# Control & Robotics

## Lego®

## Robotics Invention System™

## ROBOLAB™

## New Technology

## For Pupil Learning

The text is prepared using Microsoft Word 2002 in 12 point Times new Roman with headings in 12 point Arial with double line spacing.

Contents:

# Engineering

# And

# Engineering Technology

## Draft Syllabi

## NCCA (2000)

# Electronics Core

| Content | Detail | Student will be able to.. | Applications in Society |
|---|---|---|---|
| **Practical** | | | |
| **Electronic circuits** | • Use resistors, diodes, LED's *and NPN transistors* to produce circuits.<br>• *Use Light Dependent Resistors, variable resistors or thermistors.*<br>• Circuit soldering and de-soldering. | • build circuits using a selection of the named components.<br>• use the electric soldering iron for electronic circuits and repairs. | Burglar and fire alarms.<br>Computers, televisions, CD players.<br>Electronics. |
| **Transistor Circuits** | | • *produce circuits to include a transistor.* | Transistors in the communication industry.<br>Radio, television, computers, phones. |
| **Measurement** | • *Using the transistor as a switch.*<br>• *Use multi-meter to measure voltage and resistance.* | • *use a multi-meter to measure DC voltage in a circuit.*<br>• *use a multi-meter to measure resistance of known and unknown resistors.* | Measurement of current and voltage.<br>Fault testing of electrical circuits and wiring. |
| **Support Theory** | | | |
| **Electronic Units** | • Definition of Volts, Amps, Ohms.<br>• *Definition of Ohms law and sample calculations.* | • differentiate between the different electronic terms.<br>• *calculate current, voltage, or resistance when two elements are known.* | Electronic terms & applications.<br>Integrated circuits. |
| **Sensitive circuits.** | • *The use of components such as the LDR's, thermistors and variable resistors as sensors in transistorised circuits.* | • *explain the main characteristics of variable resistors, LDR's and thermistors.*<br>• *specify one application for each component.*<br>• *draw circuit diagram with components. in place.* | Automatic lighting circuit.<br>Fire alarms.<br>Water and moisture level detectors. |

## Energy Power & Control Option

| Content | Detail | Student will be able to.. | Applications in Society |
|---|---|---|---|
| Practical | | | |
| **Design** | • The use of control devices in solving design problems. | • identify and apply control devices in solving design problems. | Alarms, automatic light switches, automatic doors. |
| **Energy** | • Working generator. | • construct a working generator. | Generator. |
| **Control device** | • *Construction of a model unit incorporating a control device.* | • *construct a control loop to control pneumatic and electronic systems.* | Vacuum cleaners, air brakes, washing machines, dishwashers, etc. |
| **Computer Interfacing** | • Demonstration of computer interfacing. | • Use a computer to power or control a device. | Heat sensors.<br><br>Robotic control. |
| **Support Theory** | | | |

| | | | |
|---|---|---|---|
| **Energy** | • Methods of storing energy, fossil fuels.<br>• The Joule as a unit of energy.<br>• Conversion of energy (Electrical to Heat)<br>Electrical energy = Watts x Time<br>**Heat energy = Change in temp x Specific heat capacity x mass.** | • list the various types of energy.<br>• calculate the amount of electrical energy required to raise the temperature of a quantity of water by a given amount. | Electrical energy in the home.<br>Central heating, coal fire, etc.<br>All types of engines.<br>Automatic doors.<br>Central heating.<br>Laundry and food storage. |
| | • Calorific value. | • explain the difference between fuels of high calorific value and low calorific value. | Fossil fuels and the environment. |
| **Work** | • Work = Force x Distance. | • define work as moving a force over a distance. | Transport systems.<br>Engines. |

## Energy Power & Control Option

| Content | Detail | Student will be able to.. | Applications in Society |
|---|---|---|---|
| Support Theory (continued)<br>**Power** | • *Mechanical power = Force (N) x distance (m) / time (s)* | • *define power as the rate of doing work, unit of power as the: - Watt = Joule / sec = Nm /* | Engines.<br>Electrical energy.<br>Fridge freezers. |

| | | |
|---|---|---|
| **Efficiency of systems** | **= Nm / s = J / S = Watts.** *sec.*<br>• *Electrical Power = volts x amps.*<br>• Engines.<br>• Electrical generator.<br>• *Expressed as: -*<br>*Useful energy, or work output*<br>*Energy input* | • *define electrical and mechanical power.*<br>• *manipulate formula to perform calculations.*<br>• *describe the engine as an energy converter.*<br>• *sketch an electrical generator.*<br>• *explain the principle of the electrical generator.*<br>• *calculate efficiency from given data with answer given as a percentage.*<br>• *list the reasons why energy output is less than energy input in various energy conversion systems.* |

## Energy Power & Control Option

| Content | Detail | Student will be able to.. | Applications in Society |
|---|---|---|---|
| Support Theory (continued)<br>**Control** | • Elements of a control system.<br>• Transducers, sensors and actuators.<br>• Feedback.<br>• Mechanical.<br>• Electronic. | • *describe a control loop incorporating input, processing and output.*<br>• list methods of input, processing and output in a system.<br>• explain the principle of operation of transducers, sensors and actuators.<br>• specify transducers sensors and actuators to suit specified applications.<br>• explain the meaning of feedback as applied to control systems.<br>• explain the use of weights in pressure control, governors in | Central heating.<br>Computer aided machining.<br>Pressure cookers.<br>Clocks.<br>Water control.<br>Steam valve.<br>Brakes.<br>Steering mechanisms.<br>Alarm systems.<br>Fire alarms.<br>Computer Aided Engineering.<br>Robotics.<br>Automatic gates and doors.<br>Operating machines.<br>Clamping work.<br>Automation. |

| | | | |
|---|---|---|---|
| | • Computer.<br>• Pneumatic. | machines, and centrifugal clutches and brakes.<br>• *describe the function of infra-red sensors, amplification and integrated circuits.*<br>• describe the use of some form of computer interface control.<br>• explain how the five port valve operates.<br>• list common applications of the five port valve. | |

## Energy Power & Control Option

| Content | Detail | Student will be able to.. | Applications in Society |
|---|---|---|---|
| Support Theory (continued)<br>**Control (continued)** | • Design. | • combine the different control systems in solving design problems. | |

Technology Syllabus

## Option: Electronics & Control

| TOPIC | TREATMENT OF TOPIC<br>***Students should learn about/to:*** | LEARNING OUTCOMES<br>***Students should be able to:*** |
|---|---|---|
| ***Electrical Measurements*** | measurement of potential, current, resistance, capacitance, frequency (*V, A, Ω, F, Hz*) | select appropriate instruments, measure basic electrical quantities, and explain in simple terms what they have measured |
| | indirect measurement of power | calculate power in whole circuits *and in components such as resistors and motors* |
| ***Components and Circuit Design*** | resistors, capacitors, *inductors*, diodes, transistors, *voltage regulators*, *photoresistors*, *photodiodes*, LEDs, *phototransistors*, variable resistors, potential dividers and potentiometers, relays | use, *and understand the function of*, components in circuit design<br>use a potential divider as a volume control |
| | the design, assembly and testing of circuits | design, test and assemble circuits, or circuit sub-units, making appropriate use of the listed components |
| ***Power Supplies and Safety*** | select a suitable power supply for a specified application | choose appropriate power sources for selected tasks |

| **TOPIC** | **TREATMENT OF TOPIC**<br>***Students should learn about/to:*** | **LEARNING OUTCOMES**<br>***Students should be able to:*** |
|---|---|---|
| ***Electric motors*** | the mode of operation of the DC motor; back EMF; the variation of current requirement with the load | *measure the efficiency of an electric motor* |
| | reversing a DC motor | assemble switches and motors to achieve forward and reverse motion |
| ***Assembly of Pre-designed Circuits*** | assembly of bistables and astables, amplifiers, *assembly of oscillators, timing circuits* | construct and make appropriate modifications to circuits, based on circuit diagrams |
| | printed circuit boards (PCBs) | use an appropriate method to produce PCBs for a given circuit |
| | *production of prototype and commercial batches of PCBs* | *understand the production of both prototype and commercial batches of PCBs* |
| | use of prototyping boards for initial assembly and testing of circuits | demonstrate the use of prototyping boards in assembly and testing of circuits |
| | *operational amplifier circuits (op-amps)* | *design, assemble, test and modify operational amplifier circuits using integrated circuits for signal amplification, level detection and voltage comparison* |
| ***Sensors*** | sensors for sound, heat, light (photoresistive and photovoltaic), movement | design, assemble, test and modify basic sensors |

| **TOPIC** | **TREATMENT OF TOPIC**<br>***Students should learn about/to:*** | **LEARNING OUTCOMES**<br>***Students should be able to:*** |
|---|---|---|
| ***Logic Circuits*** | basic logic gates: AND, OR, NOT and NAND<br>truth tables | construct truth tables for up to four inputs using an array of up to four logic gates |
| | combinations of gates | combine logic gates appropriately using ICs |
| | the main logic families (TTL and CMOS) | select the appropriate type (CMOS or TTL) of IC for a particular task |
| | the use of logic gates with sensors and output devices | design, construct, test and modify simple systems using sensors, combinations of gates and output devices |
| ***Inputs and Outputs*** | buffers (transistors, *amplifiers, paralleled outputs*)<br>Schmitt trigger<br>binary inputs | design, construct, test *and modify* buffer or driver circuits for a variety of output devices<br>use gates with Schmitt trigger inputs to sharpen digital signals |
| ***Counters*** | clock circuits, de-bouncers, counters, seven segment displays and display drivers | design, construct, test and modify simple counting circuits capable of counting inputs from switches or clocks |

## Option: Applied Control Systems

| TOPIC | TREATMENT OF TOPIC<br>***Students should learn about/to:*** | LEARNING OUTCOMES<br>***Students should be able to:*** |
|---|---|---|
| ***Robotics*** | classify robotic joints by degree of freedom and co-ordinate frames | identify robot types<br>specify robotic joints for particular applications |
| | do simple calculations of forces and moments involved | size and specify robotic structure and arms |
| ***Introduction to Robotic Control*** | classify robots by structure and application, with emphasis on manufacturing applications | be aware of industrial applications of robotics<br>*identify suitable robotic structures and configurations for specified tasks* |
| | | construct block diagrams of simple robotic controllers; calculate gains |
| | *understand the principles of open and closed loop control* | |
| | understand the principles of operation and control of dc servos and stepper motors | select DC servos or stepper motors and controllers |
| ***A/D and D/A Conversion*** | *analogue to digital and digital to analogue converters (A/D and D/A)* | *incorporate and use digital inputs and appropriate A/D converters*<br>*incorporate D/A outputs where appropriate* |

| **TOPIC** | **TREATMENT OF TOPIC**<br>***Students should learn about/to:*** | **LEARNING OUTCOMES**<br>***Students should be able to:*** |
|---|---|---|
| ***Control*** | use of computers or other programmable devices (such as PLCs or PICs) to control various devices | use a programmable device to control circuits or sub-assemblies which they have constructed, e.g. LEDs, bulbs, seven-segment displays, DC motors |
| ***Programmable Devices*** | understand the principles of combinational and sequential logic | program simple logic sets using Grafcet or function diagrams |
| | programme a robotic device to carry out specified operations and to sense functions | programme a robotic device to do specified tasks *and to modify outputs in response to sensed conditions* |
| ***Pneumatics*** | identify pneumatic actuators, directional control valves, pressure and speed control valves, stop valves, service stations | use appropriate calculations and/or charts to facilitate the selection of circuit elements |
| | *understand the principles of selecting control strategies, e.g. totally electronic, electro-pneumatic, totally pneumatic circuits* | design, cost and build single *or mixed* technology circuits |

## Module 1: Electronic/electricity introduction

Aim:

To reacquaint pupils with electronic and electricity concepts

Objectives:

At the end of this module pupils will be able to:

1. Describe the different terms.
2. Use Ohms law for calculations in theory examples and practical work.
3. Identify the different signal and current types.
4. Design circuitry for project work using the new learning.

Evaluation:

This material is a revision of Junior Certificate concepts. Practical work with simple circuits can be a good method of revisiting core concepts.

## Module 1: Electronic control

It is expected that most pupils will have completed science to junior certificate level. This module will revise key areas of particular interest to electronic control as an introduction to applied control systems.

Voltage

The pressure of the flow of electrons is called voltage

If the flow of electrons is high the voltage will be high.

Voltage may also be called a potential difference.

Voltage is measured in volts (V) using a voltmeter over the component to be measured.

Current

The rate of flow of electrons in a circuit is called the current.

Current will only flow when there is a voltage (potential) difference.

Current flows form high potential difference to low potential difference.

Current is measured in Amperes (I) using the millimetre integrated into the circuit

Resistance

A conductor's property, which controls how much current flows.

The larger the resistance a conductor has the less flow of electrons (current).

Add the sum of resistors in series circuits to get the resistance.

In parallel circuits use R1xR2 / (R1 +R2) to get the resistance

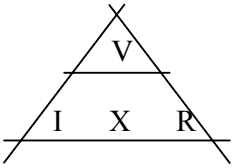
Resistance is measured in ohms (Ω).

Ohm's law

Current, resistance and voltage are related and can be combined using Ohm's law

Voltage (V) = Current (I) X Resistance (R)

Ohm's triangle used as an easy way to remember.




Power

The rate at which a resistor produces heat energy per second is called power.

Power is equal to the voltage X current

voltage ^2 X resistance

current ^2 X resistance

Power is measured in watts (W)

Electrical signals

Analogue signals are continuous.

Digital signals are discrete.

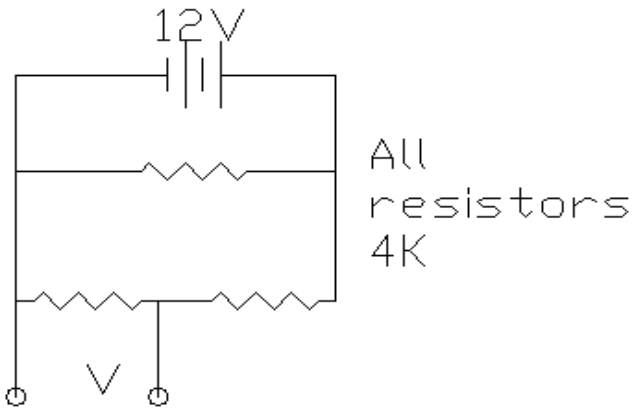
Electrical signals are easy to process using inexpensive circuitry and are reliable. Signals are easily transmitted over large distances and stored. What method does a mobile phone text message use?

Direct and alternating current

Electronic projects need steady sources of electricity; therefore direct current is used in the form of batteries. Alternating current has a flow in current direction constantly. This is measured as frequency (F)

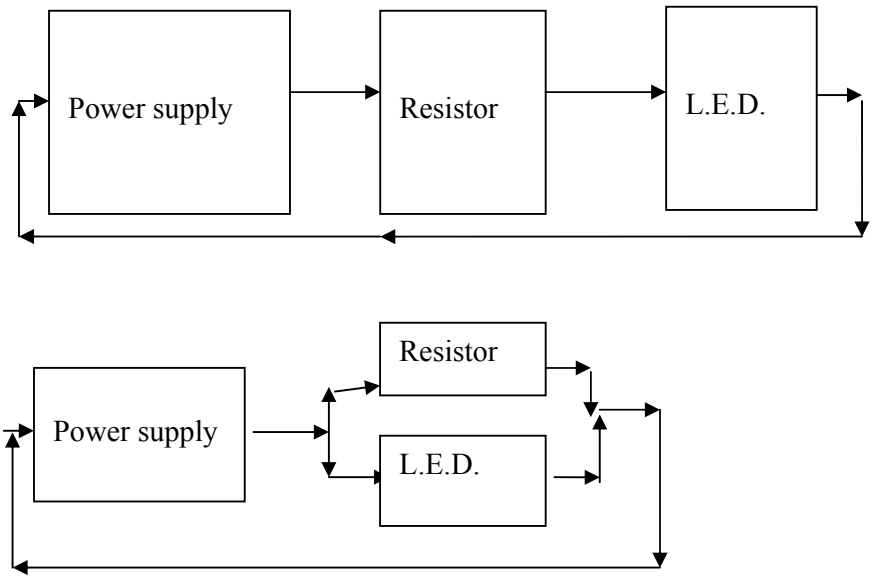
## Module 1: Task Sheet

1. If two 5KΩ resistors are connected in parallel what is the combined resistance. What is the current in the circuit if three 1.5 volt batteries are connected in series?
2. What is the combined resistance if two 10KΩ resistors in parallel and a 5KΩ resistor in series with them. Calculate the current between them (two readings).
3. A 7K4 and a 5KΩ resistor are connected in series. What is the total resistance in the circuit? Why is this resistance not equal to the readings using a physical experiment? Why are physical experiments important? What is the voltage between both resistors if 8 volts is passed between the circuits?
4. What is the reading of the voltage in the circuit below?




5. Since electric current is a flow of charge why are two wires used to conduct current instead of one.

6. In wiring up the simple circuit below which method is correct to prevent the L.E.D. from receiving to much voltage? Explain why?




7. A 220 V electric soldering iron draws a current of 20A. What is the internal resistance of the soldering iron?

8. If light bulbs are connected in series or parallel which arrangement will provide best illumination? Show why this is the case and refer to the power usage.

9. Calculate the current in a 150 Watt bulb when it is operated at 110V and 220V. (two readings)

## Module 2: Electronic systems

Aim:

To introduce pupils to electronic systems including signals and digital logic, involving truth tables and Karnaugh mapping

Objectives:

At the end of this module pupils will be able to:

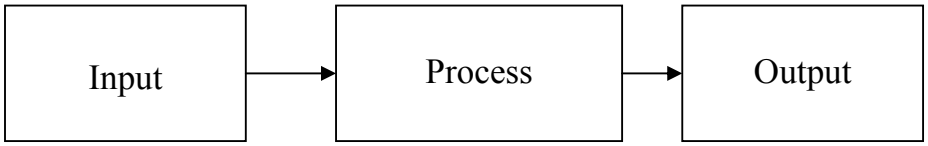
1. Explain the difference with electronics and electronic systems.
2. Contrast analogue and digital signals, demonstrate a signal.
3. Construct truth tables for problems and draw logic circuits solving the problem.
4. Describe what digital logic is using diode logic.
5. Use Karnaugh mapping to help solve problems with many variables.
6. Combine logic gates to arrive at different outcomes (combinational principals).

Evaluation:

The task sheets will provide help in checking to see how pupils are progressing. The truth tables and logic statement are adequate to solve many problems if Karnaugh mapping proves difficult. The class room examples will give you an idea of how the topic is progressing by pupil interaction.

## Module 2: Electronic systems

An electronic system is an arrangement of electronic components with a defined set of inputs and outputs. They take in information, process this and produce an output. A typical example of an electronic system is a robot. The difference between electronics and electronic systems is that electronics studies individual components and is component centred while electronic systems studies a range of components in a system that perform a particular function.




### Electrical system diagram

An electronic system is the study of a range of components working together to perform a function which is divided into system blocks and sub blocks.

The information transfer flows in the same direction as the arrows. This information is in the form of an electrical signal (current). Electrical systems are broken down into subsystems (i.e.) input, process and output. Input subsystems usually convert information from the environment into electrical form. Touch and light sensors are examples of input subsystems. The processing subsystem then takes this signal and can modify it or combine it with another signal, depending on the process. The output subsystem then takes this signal after is has being processed and converts it to another form of energy, buzzer of motor.

## Signals

Signals have being introduced earlier, now we are going to discuss them more.

Most physical quantities vary in continuous matter. They change smoothly from one value to another and can have an infinite number of values.

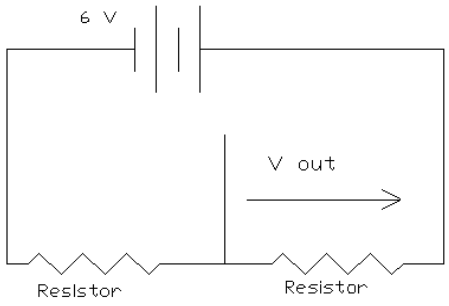
Analogue signals have relationships with physical properties. For example a light sensor reads a value of 45% on a black surface. So the signal has a maximum and a minimum value.

Some qualities do not change smoothly and change abruptly between a number of fixed values. In electronic systems a varying physical quantity is represented by an electrical signal that varies in a manner, which describes that quantity.

Digital signals are encoded. Binary digits are the simplest form of digital signal. These signals are of importance as they are widely used in electric logic circuitry. Analogue signals can be converted into digital signals when a threshold is passed in the process subsystem. The signal is then forwarded to the output subsystem as a digital signal.

Digital signals are either true (1) or false (0) so they are limited. Analogue signals have a wide range of possibilities and the different values can produce different out puts.

The simplest form of signal is achieved form a potential divider circuit.



## Logic

Logic is described as reasoning, thinking about something at a clear level.

Logic can be based on statements or by using binary numbers.

Logic involving statements is called analogue, an example of this logic is.

> "John will eat his dinner and watch television, but will
>
> not watch television and do his homework together"

Logic based on binary numbers 1 and 0 is digital. This logic can be completely described in digits. There is no in-between as may be the case with analogue. A bulb in a closed circuit is either on (1) or off (0); there is no half on or half off.

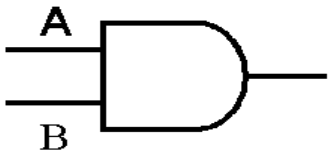
Digital logic has rules that use reasoning in a structured way to consider all situations possible.

Due to the fact that digital logic uses 1s and 0s Boolean algebra can be used to solve problems, however this can become quite complicated and over 15 laws need to be learned. Therefore we are going to use truth tables and Karnaugh mapping to find out what our circuit should look like.

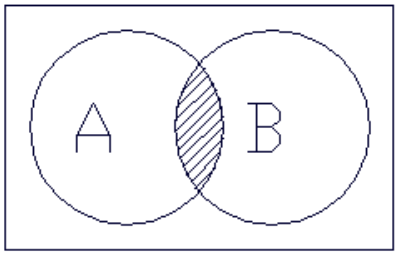
Logic gates are used in digital signal processing. A gate is a controlling device that may be either open (1) or shut (0). We will discuss the five major logic gates using Venn diagrams and truth tables. Truth tables are the method in which all the possibilities are taken into consideration. Logic gates have standard symbols and these are shown overleaf. (American standards)

## Logic Gates

### AND Gate



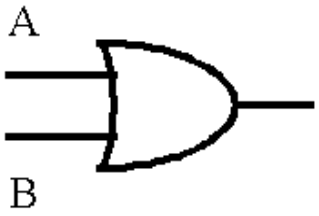

Both A **AND** B inputs need to be binary 1 to give an output 1. An example is two switches in series, both are needed to turn on the light.

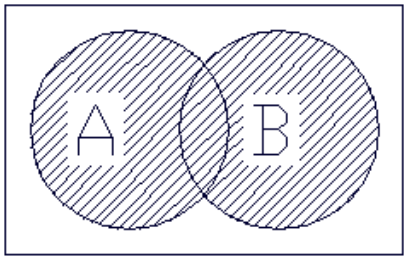



| B | A | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### OR Gate

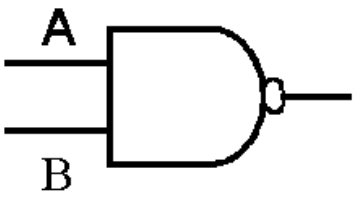
Either A **OR** B inputs need to be of binary 1. An example is two switches in parallel, where either A or B will turn on the light.



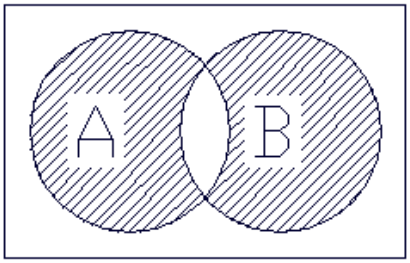




| B | A | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## NAND Gate



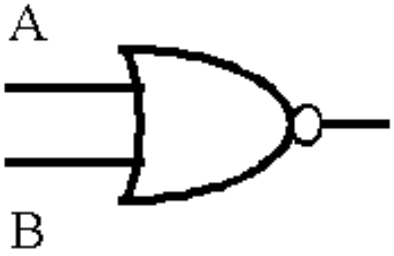

A NAND gate is equivalent to an AND gate followed by a NOT gate. The output is (0) if both inputs are (1) Otherwise, the output is (1).

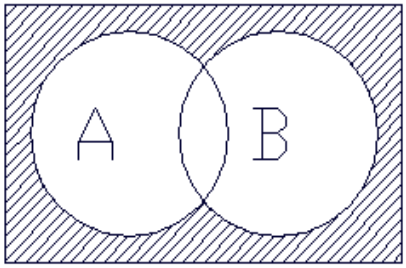



| B | A | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## NOR Gate

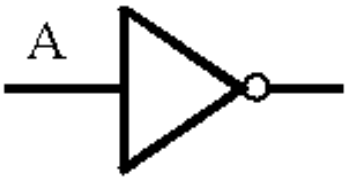
A NOR gate is equivalent to an OR gate follow by a not gate. Its output is (1) if both inputs are (0). Otherwise, the output is (0). (Neither gate)



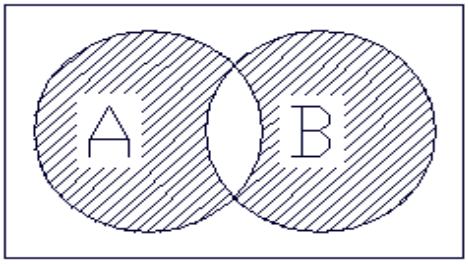




| B | A | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## NOT gate




A NOT gate (inverter), has only one input. It reverses the logic state, which is the polar opposite of the input signal.




| B | A | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

### Combinations of gates (Combinatory Logic)

When circuits are to be assembled using logic gates, it is important to note that when different gates are combined there outputs can change. It is also possible to create gates in you circuit from other gates that may not be at your disposal. An example would be that you want to use an OR gate but you only have NAND, AND and NORS available to you. What can you do? Below are some commonly found rules for using logic gates?

If a **NOT** gate is used after an **AND** gate the result is the same as a **NAND** gate.

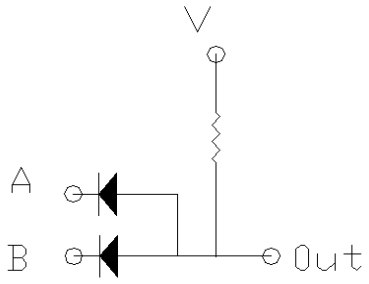
If a **NOT** gate is used after an **OR** Gate the resultant is the same as a **NOR** gate.

Two **NAND** gates in succession produce the same output as an **AND** gate.

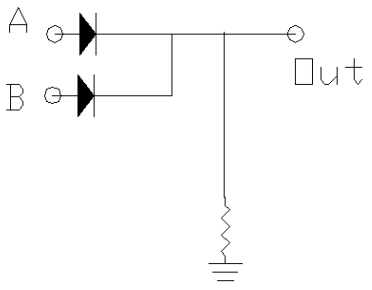
Two **NOR** gates in succession produce the same output as an **OR** gate.

Logic gates can be made form diodes (Diode Logic). Diodes only conduct electrial current in a single direction. Therefore the diode can be used as an electroniuc switch.

The diagram below shows an AND gate. A resistor is used to pull the output voltage up to logic (1). If both A, B are at logic (1) unconnected then the output will be at logic (1). Let's say that input A is grounded logic (0). Then the A diode will conduct producing a Logic (0). A **AND** B must be logic (1) in order for the output to be logic (1).



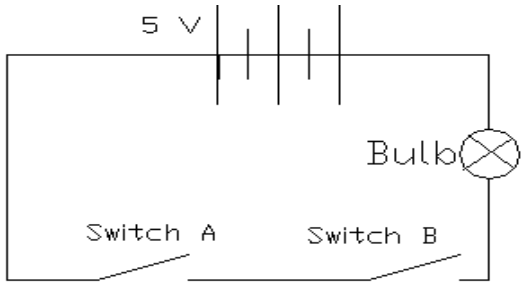

The diagram below shows two diodes with inputs A, B. Notice that the diodes are reversed. This is a logic OR gate. Let's say that logic (1) is represented by 2 Volts and logic (0) is represented by the ground terminal. If A, B are both at logic 0, the output will be 0 volts. However if A OR B is raised to 2 volts, its diode will become forward biased and current will flow through it. This in turn will force the output up to logic 1. If A, B is logic (1) the output will still be logic (1). The gate is a logical OR function.



## Truth tables

Truth tables are used as a method of finding out what gates must be used and how they should be constructed. Binary digits are used and they are 1 and 0 (on/off). Every possibility is recorded in the table.

For example a circuit with two switches in series (input A and B) and a lamp is shown below.




Switch open = 0

Switch closed = 1

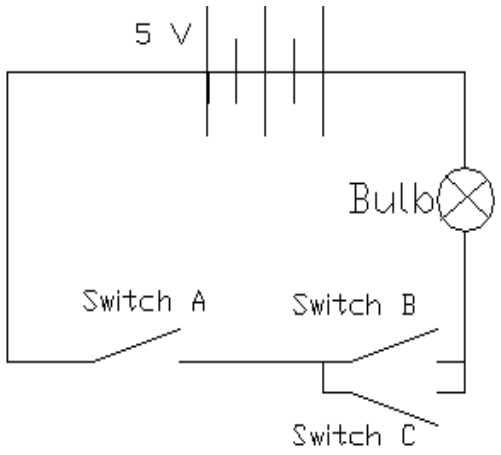
Fill in the truth table now when you understand the problem. What is the symbol for the gate you should use?

| Input B | Input A | Output Bulb |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

To ensure that you have considered every option/possibility you count the number of rows down and it should be equal to the number 2 to the power (number of inputs). The same can be done for two switches in parallel circuits.

This can become more complex with 3-4 inputs and the gates to use will not seem as familiar.

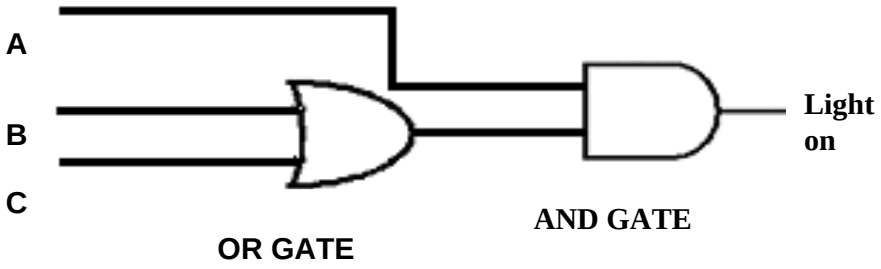
Consider the circuit below and construct a truth table for it including the circuit diagram.



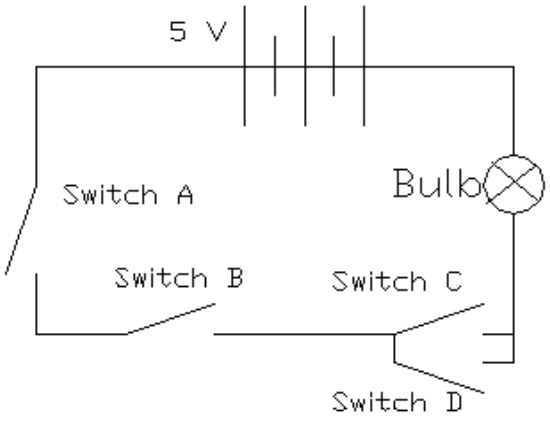

There are three inputs A, B and C. the truth table will contain 2^3 = 9 lines.

| A | B | C | Bulb On |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

From the diagram and truth table it is clear to see that the bulb only lights for three of the conditions. The A switch has to be at 1 for the light to work plus either B or C need to be at 1 for the Bulb to light. Therefore from this information a logic circuit diagram can be constructed.



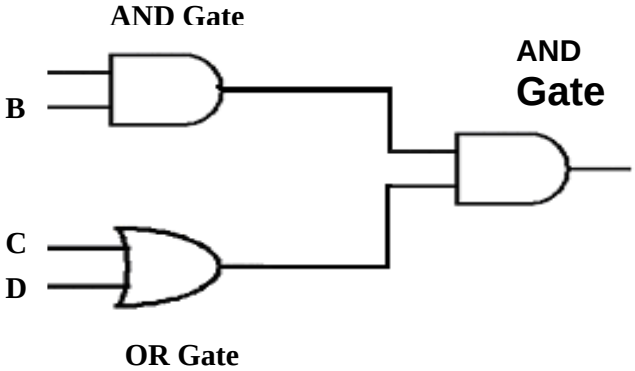
Consider the circuit below; construct a truth table and a logic diagram.




There are four input switches and one light bulb. The truth table will have 2^4 variables.

| A | B | C | D | Bulb On |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

For the bulb to light both A and B need to be 1 and either C or D needs to be 1.



We will construct another more complex truth table with three inputs. The condition is that one of two inputs is to pass its data directly to the output. Input A is the selected line therefore if A is 1 then C is selected and if A is 0 then B is selected.

To construct the truth table you will need to understand the above.

| Input A | Input B | Input C | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Is every possibility considered 2^3 = 8

It can be seen from the above table that when A = 0 then the output = C and when A = 1 then the output = B.

It is impossible to consider what gates to use therefore we need to analyse our information further using a Karnaugh map. This is an ordered array of possible combinations. The Karnaugh map is called a map, as it should be considered as cylinder. Its construction for tree inputs is as follows.

| | *A* | | A | |
|---|---|---|---|---|
| *C* | ABC | AB*C* | A**B***C* | ABC |
| C | *A**B***C | *A*B*C* | A**B**C | A*B*C |
| | *B* | | **B** | *B* |

A logic statement is required before we can fill out this map. The statement is the conditions that need to be achieved when the output is one. Therefore when the output is 1:

"A is 0 and B is 0 and C is 1 or A is 0 and B is 1 and C is 1 or A is 1 and B is 1 and C is 0 or A is 1 and B is 1 and C is 1"

Gates can be implemented now but we would need eight **AND** gates and three **OR** gates. Therefore this can be simplified using the Karnaugh map.

To convert the construction into binary digits the underlined *italic* letters are equal to 1. The output should be recorded in their space instead of them. For example the first box reads *ABC* is equal to 1,1,1, while *A***B**C is equal to 1,0,0.

So look over to the output line and record either 1 or 0.

The solution is

| | *A* | | A | |
|---|---|---|---|---|
| *C* | 1 | 0 | 1 | 1 |
| C | 1 | 0 | 0 | 0 |
| | *B* | | **B** | *B* |

The result of the Karnaugh map is

C x *A* + **B** x A

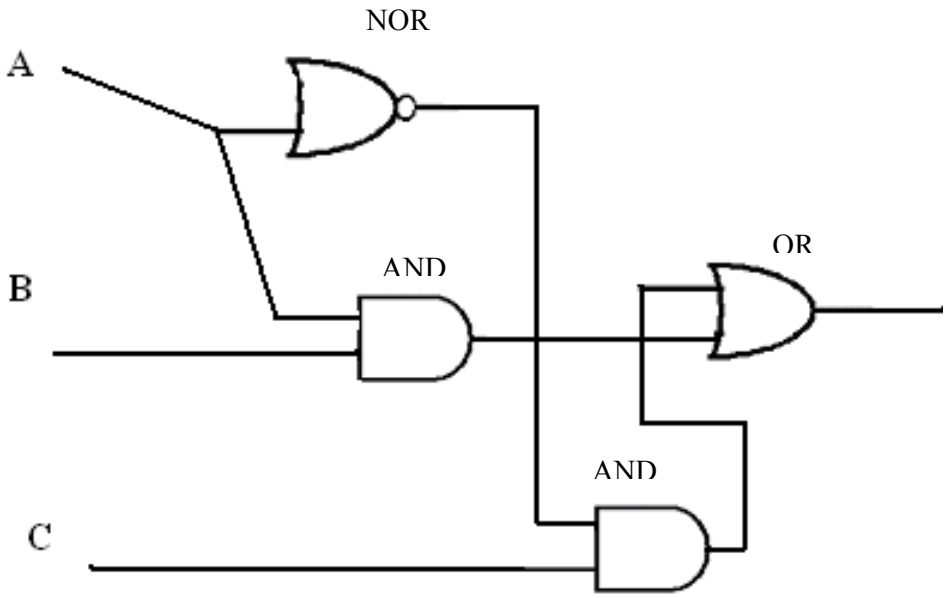
Rules to follow are

+ = OR gate x = AND gate *Underlined and italic* = inversion.

The diagram solution should look like this

The logic can be put in a statement A *AND* B (1) with inverted A *AND* C (2)

With (1) *OR* (2)



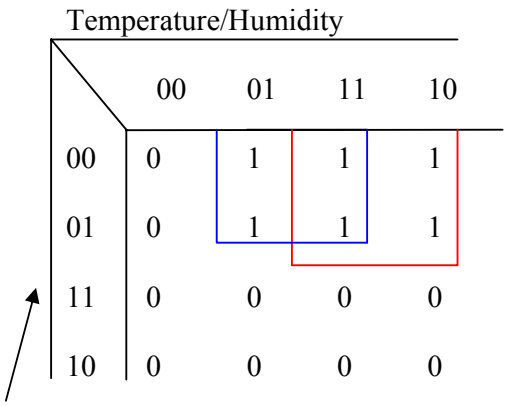
A practical application of a digital logical circuit is now demonstrated.

The window of a green house is needed to open automatically. There are four input sensors used and a binary digit is given for what state they are in. Draw the logic circuit.

| | |
|---|---|
| 1. Temperature sensor, | 0 if temperature is less than 30 degrees |
| | 1 if temperature is greater than 30 degrees. |
| 2. Humidity sensor, | 0 if humidity is less than 50 % |
| | 1 if humidity is greater than 50% |
| 3. Wind speed sensor, | 0 if speed is less than 20 MPH |
| | 1 if speed is greater than 20 MPH |
| 4. Window sensor (Opened or closed) | 0 if window is opened |
| | 1 if window is closed |

| 1 | 2 | 3 | 4 | Open window |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

The Karnaugh map is used to organise the solutions. There are four variables now so we cannot use the previous example. We will need to construct our own Karnaugh map taking all the variables into consideration. The method that I am taking is Temp/Humidity against Wind speed/window.




Wind speed and Window

From the Karnaugh map you find six 1's together making up two squares. Two equations can be retrieved from this map.

The blue box is humidity high by wind speed inverted, and the red box is temperature hot by wind speed inverted. The wind speed is inverted for both, as it equals 0 on both equations.

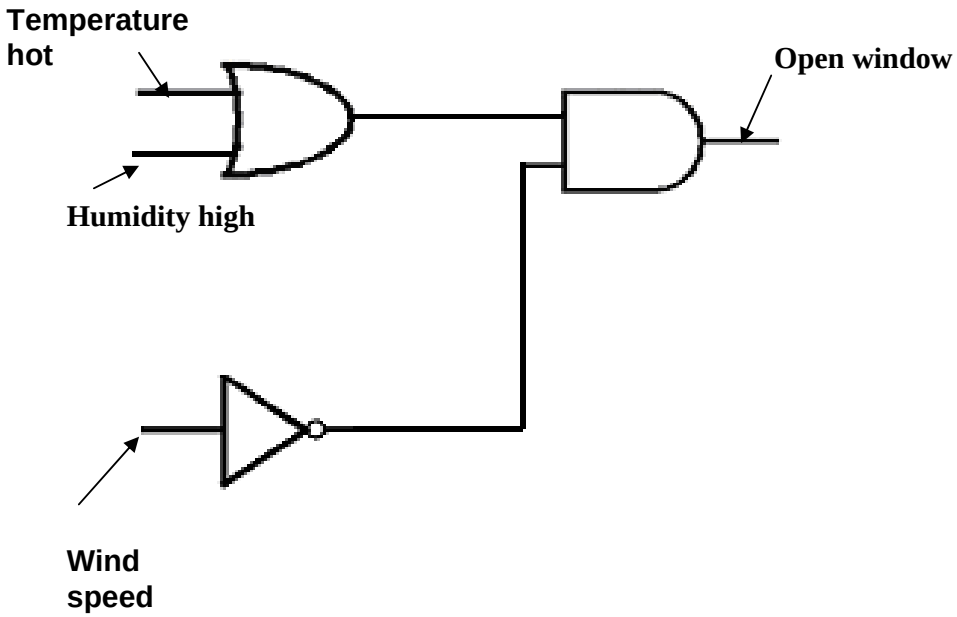
Therefore when the both of the equations are combined they equal

= wind speed (inverted) by x (humidity high + temperature hot)

Open window = *wind speed* x (humidity high + Temperature hot)

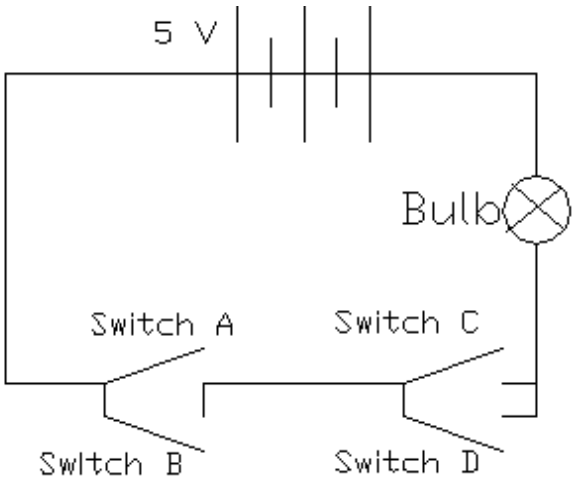
Logical statement:

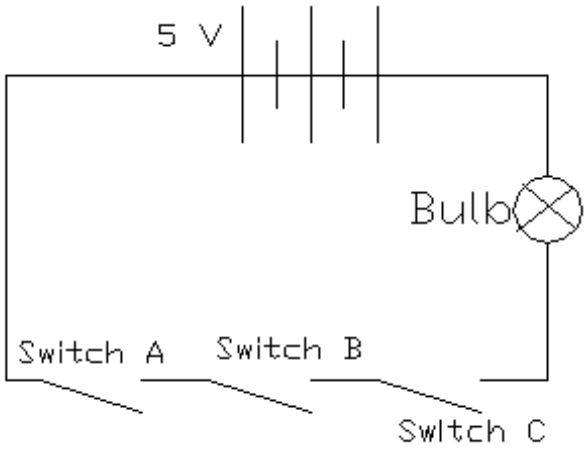
From the logic statements we need 3 gates an OR gate, AND gate plus INVERTER gate.

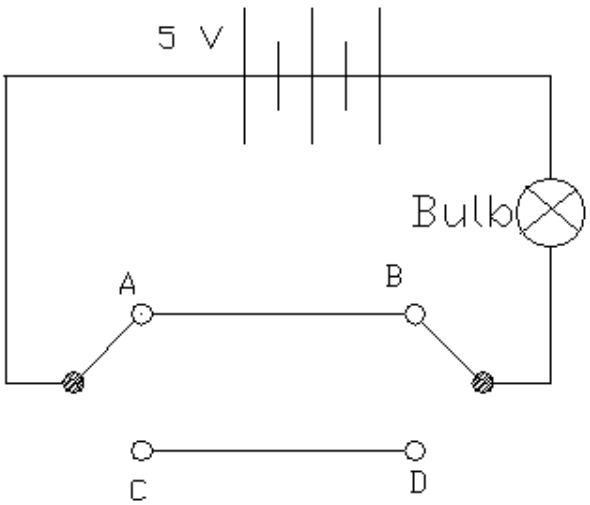


## Module 2: Task sheet

1. Draw truth and logic diagrams for the following circuits.



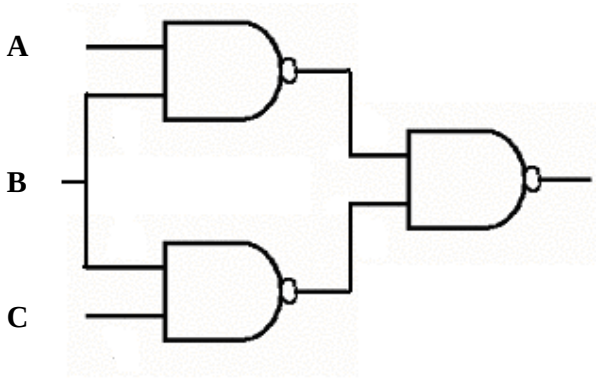



2. A logic system needs to be designed for a safety feature on a hot liquid bath for cleaning sheet metal.

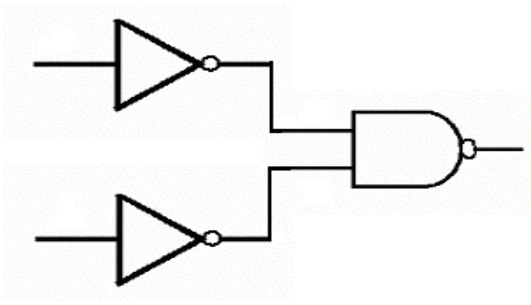
   The system has a gas blow torch heating the tank. There is two sensors (a) flame sensor (b) temperature sensor. The tank needs to be kept at 50 degrees, if the temperature gets higher or if the flame goes out an alarm will need to be triggered. The only constraint is that the alarm needs logic 1 to operate.

   Draw both a logic diagram for the solution and a electronic systems diagram.

3. Draw a truth table for the combinational logic set up.




4. What gate is similar to the action of NAND gate followed by a Not gate.
5. What gate gives the same output as arrangement of gates below?



## Module 3: Digital signal processing

Aim:

To develop pupils electronic knowledge base by discussing the main logic families.

Objectives:

At the end of this module pupils will be able to:

1. List the different logic families
2. Select a specific family for a particular circuit.
3. Complete the accompanied task sheet.

Evaluation

This module is evaluated by pupils work on the task sheet. Their independent research and circuitry building should be increased here.

## Module 3: Digital signal processing

Logic families

Integrated circuits (ICs) are small circuits integrated into a plastic chip. Two prominent logic families exist they are the TTL (Transistor Transistor Logic) and CMOS (Complimentary Metal Oxide Silicon). What type to use? Both families have advantages and disadvantages.

TTL

TTL chips use a narrow range of voltage form 5 volts to 4 volts. This voltage is not achievable with two AA (1.5V) batteries. TTL chips are quick at responding to signals. TTL has poor resistance to electrical noise. Electronic noise can appear as highs and lows in a circuit and can affect the signal causing false signals to be responded to by the IC. Protection may be needed for these TTL ICs and is provided by with the use of capacitors. TTL inputs when not in use on the IC act as a logic (1). This needs to be taken into consideration when designing circuits. TTL chips have the same threshold levels and this allows them to be used in mixed integrated circuits.

CMOS

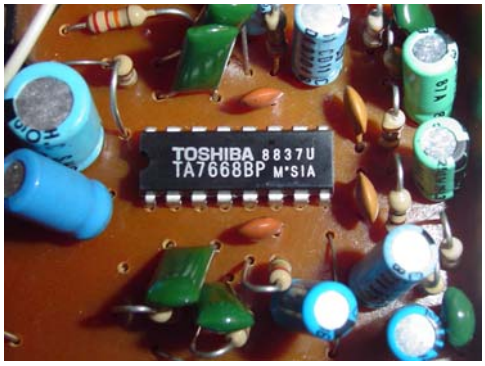
CMOS logic is the latest of the digital logic families. The operating voltage has a larger range between 2.5V – 15V. This makes these gates useful for battery-operated electronics. CMOS chips take considerable time to process signals. CMOS is susceptible to static electricity and can

render the chip useless. The pins on an IC should never be touched to avoid this. The operator and work areas should be grounded. CMOS is immune to circuit noise and no precautions are needed. A CMOS unused input has to be connected to either logic (1) or logic (0). If this is not carried out the circuit will malfunction.

Today these families are being replaced by HC and HCT. They have similar numbering to TTL but are manufactured on CMOS technology.

What family to use is a common question that designers and technicians need to ask themselves? The above information is helpful but common sense needs to be used also. For example you should use the same logic family as the ICs logic. Remember that chips from one family are not interchangeable and do not carry out the same functions as chips from another family.

ICs are shown below they are from a radio and a touch phone. You can see the numbers on the chips. Also note the notch on the left hand side of the chips these are present to indicate where the first pin is.





There is various hand made logic also.

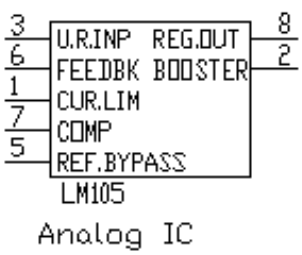
There is the:

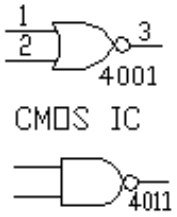
DL (Diode Logic) that we built the **AND** and **OR** gates from. However it is impossible to manufacture **NOT** gates from diodes.

DTL (Diode Transistor Logic) the diodes are used as above and then the transistor acts as an inverter.

RTL (Resistor Transistor Logic) use transistors that amplify invert and combine signals. However they use quite high level of power. They are slow to react to signals.

The AutoCAD programme has pre-programmed integrated circuit logic gates. It contains both Analogue and CMOS drawings as can be seen below.




Commonly used Logic gates and codes.

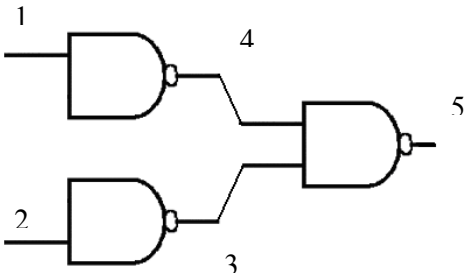
2-input **NAND** gate= 74HC00 2-input **NOR** gate= 74HC00

2-input **AND** gate= 74HC08 2-input **OR** gate= 74HC32

**Inverter**= 74HC00

## Module 3: Task sheet

1. What logic family would you use if building an **OR** gate for a project.
2. Draw an **AND** logic gate using two diodes and a resistor.
3. How could you turn this into a **NAND** gate?
4. Complete the following truth table.




| Inputs | | Outputs | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | | | |
| 0 | 1 | | | |
| 1 | 0 | | | |
| 1 | 1 | | | |

What logic gate will perform the same function as the arrangement above?

5. Explain the function of the **NOR** gate using the Venn diagrams and how could it be made not using a **NOR** gate.

6. Wearing safety belts in cars is now law, design a logic circuit that will set off a flashing LED only when there is weight on the set, and the ignition is on. The LED should not be on when the seat belt is on.

7. Design a logic circuit that enables a metal lathe to be operated when either manual switch A or B and safety visor C are pressed. But if switch D is pressed telling you that the work needs to be tightened. A logic statement a truth table and a Karnaugh map is necessary. An electronic circuit diagram using switches as logic gates must also be drawn.

## Module 4: Transistors

Aim:

To reintroduce the transistor and its fundamental uses in electronic systems

Objectives:

At the end of this module pupils will be able to:

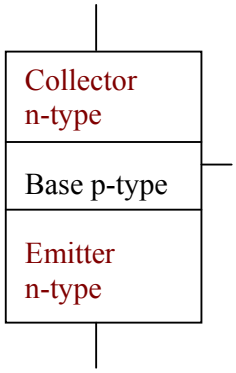
1. Describe and illustrate how a transistor works.
2. Compare the two types of transistor.
3. Use transistors as amplifiers.
4. List the principles, construction and operation of the noninverting amplifier, comparator, Schmitt trigger and buffers.

Evaluation

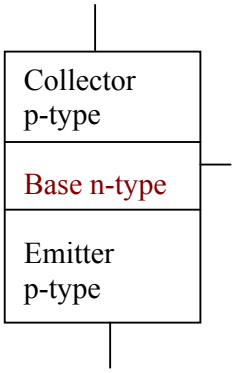
This is quite a large module with quite a lot of different systems made from the same components. Each system should be demonstrated separately and at the conclusion integrate then together in a larger system. The task sheet will give you some idea where pupil level of understanding is at. It is important that the understanding of each component is not required a systems approach is required.

## Module 4: Transistors

The name transistor comes from transfer and resistor. The diode is known as a passive device as it is impossible to change its effective operation. The transistor is an active device as its operation can be controlled by a signal. Bipolar junction transistors are the most common used especially in amplifiers. The transistor is composed of three layers of semi conducting material. The semi conducting material is referred to as p-type or n-type. They can be made with either the n-type or the p-type in the centre. The thin central layer is called the base while the ticker outer layers are referred to as the emitter and collector.




npn transistor



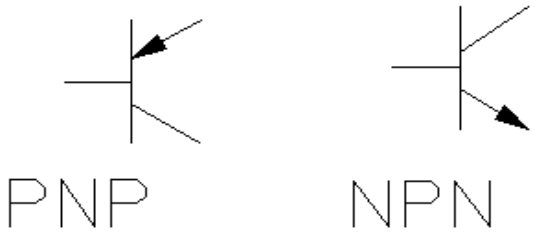

pnp-transistor

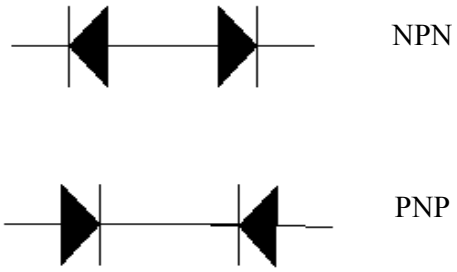
The transistor has two junctions the first between the emitter and the base and the second between the collector and the base. For a pnp-type transistor the emitter to base junction is reversed biased causing high resistance. While the collector to base junction is forward biased leading to low resistance.

Bipolar junction transistors are resistors whose resistance decreases when the base current increases and whose resistance increases when the base current decreases. Therefore the base current controls the larger collector current. A small change in the base current leads to a far greater change in the collector current. Hence the transistor can control and vary a signal but it can also

amplify it. To prevent damage the base current needs to be small therefore a large resistor is used in circuits involving transistors. Another resistor is also used to make sure that the collector current is not too large, this is called a load resistor.

Here are two of the common transistor type's icons from AutoCAD.



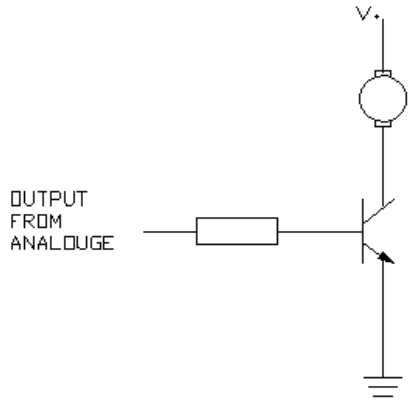

Transistors resemble diodes connected back to back as shown above. The resistance between the emitter and the collector base can be measured using the ohmmeter; the leads may need to be reversed as with the diode.

Transistors can be used as either amplifiers or switches. Using the transistor as a switch is simply on/off. Using the transistor as an amplifier is more complex.

## Current amplification

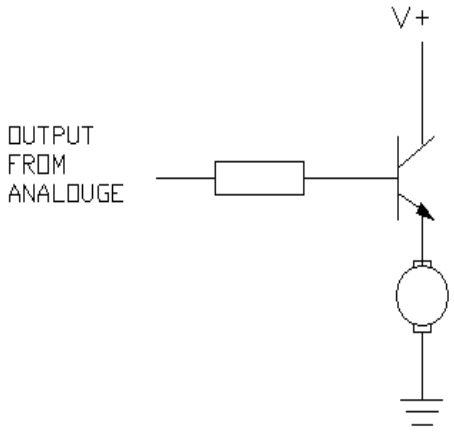
When a signal reaches a system is has limited energy. The power needs to be boosted to produce an output. A transistor arrangement is used to do this amplification.

The common emitter is used to turn on circuits that need a higher current. A small input voltage can saturate the transistor creating a larger current.




The motor is connected to the positive terminal. The current gain C*g* = base current by the collector current.

The transistor emitter follower method is used when proportional current amplification is needed. In this circuit the motor is ground (emitter side of the transistor). A greater current amplification is possible with this set up.



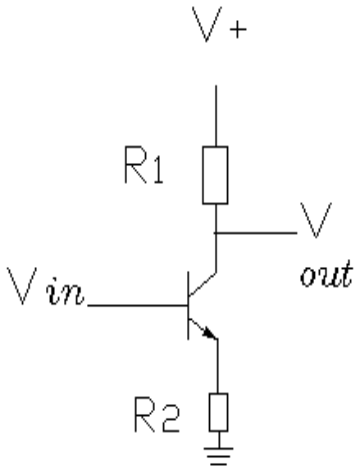

The advantage of the emitter follower is that the current across the motor is less than the current applied to the base.

## Transistors as amplifiers

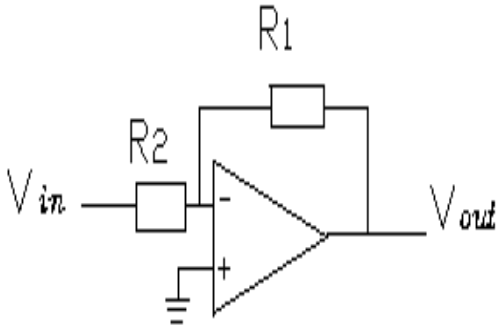
Analogue signal processing (voltage)

The two most common types of amplifiers for analogue signals are inverting and non-inverting. Amplifiers do not change the signal they just make it stronger. An amplifier is drawn in circuit diagrams similar to a **NOT** gate without the circle. Electrical noise and resistance in the circuit can corrupt signals. To rectify this problem a transistor amplifier is used.

The inverting amplifier



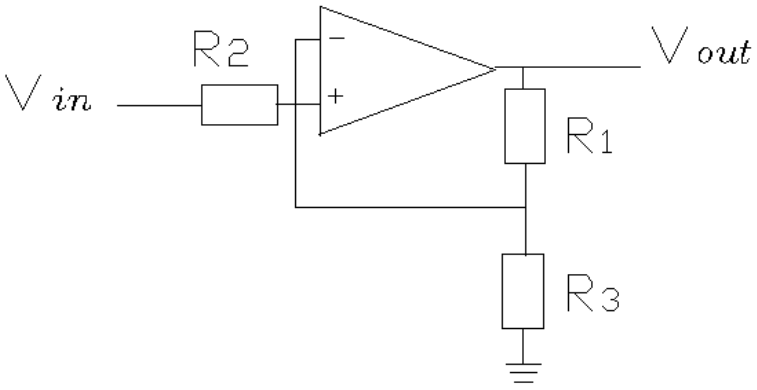

Transistor based amplifier




op-amp

The voltage *GAIN* in the inverting amplifier above can be estimated by R1/R2. This is inverting therefore a minus is used in the *GAIN* equation. High gain levels can be achieved but this needs to be kept low due to distortion. It is important to note that the signal gain is achieved by the resistors and not from the op-amp.

**The non-inverting amplifier**



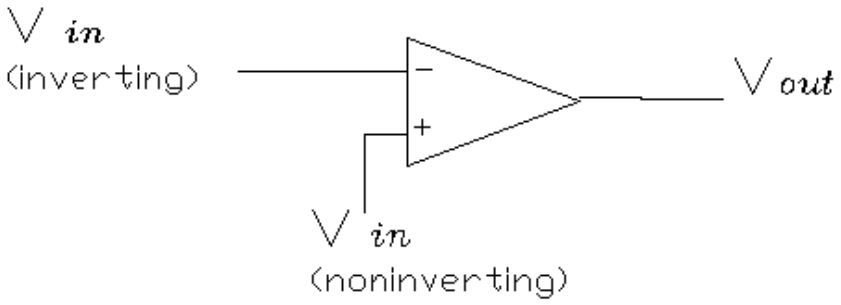

The standard non-inverting amp is shown. The *GAIN* in signal is equal to the **V**out / **V**in. The output signal is in phase with the input signal. A non-inverting analogue buffer can be manufactured from the above circuit if R1 is replaced with wire and if R3 is removed. A buffer will be explained later in relation to current amplification.

Signals may need to be combined or mixed together. It is not possible to place the two signals/inputs in to a system. Signals may be mixed for example the karaoke machine uses an amplifier called the summing amplifier that of adding together the signals. To irradiate electronic noise in the system a difference amplifier is used. The difference amplifier really separates the signals out and only lets the signal selected be herd/processed.

Above we have being dealing with analogue signal progressing. But often it is better to process these signals as digital forms. For example the light sensor has risen above 30%. This is best achieved by a digital signal Yes/No. A comparator or threshold detector is used.

## Comparator

A comparator has two inputs and a single output. The input signals are both inverting and non-inverting. If the non-inverting input signal is greater than the inverting signal the output will be (1). If the non-inverting input signal is less than the inverting signal the output will be (0).

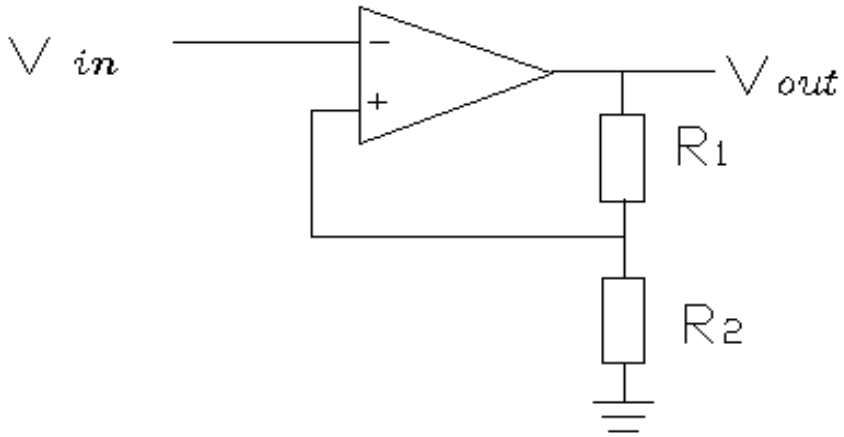



If one of the inputs is set and the other is left as a variable then the comparator can be viewed as a threshold detector as its state will only change when a certain level/threshold has being achieved. The values of the inputs are dependent on the voltage input. For example if 9 volts is being used you will need to use CMOS instead of TTL logic gates.

If the non-inverting is set at 5V and the inverting signal is alternating between 4.9 and 5.1 volts then the output will be jumping between (1) and (0). This is usually unwanted. This oscillating can inhabit the devices operation. To prevent this a Schmitt trigger is used.

### Schmitt trigger

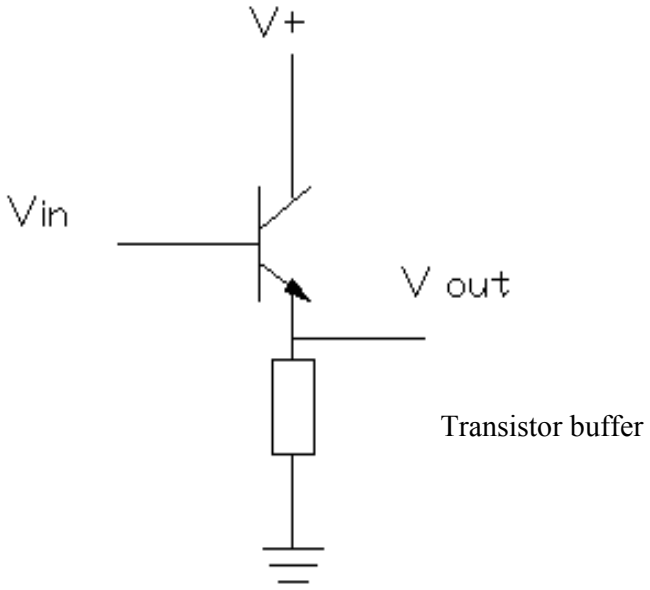
A Schmitt trigger is used to overcome this problem. A tolerance called *dead band* in electronic terms is set up to provide positive feedback to the comparator. For example a light sensor is set at 40% and then the sensor reads a value of 39% the logic (1) will not change until a difference of 4% is reached. Therefore the logic will stay (1) until 36% and then it will turn to (0). The size of *dead band* is related to the resistor values.



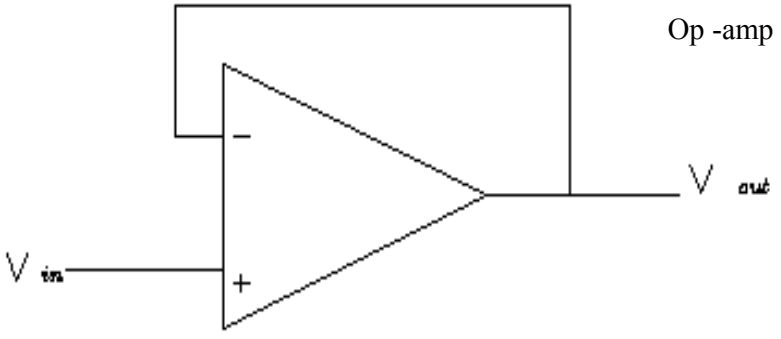

Most logic gates have Schmitt triggers integrated into their inputs; this can eliminate noise in some signals. It is important to note that positive feedback is used here. Positive feedback allows the output to stay stationary for a considerable amount of time and prevents the logic from jumping. The Schmitt trigger acts as a switch that changes its output at two different threshold levels. The circuit diagram for a Schmitt trigger is shown above.

## Buffers

Are used for specialised circuits that require extra input current, these are designed to drive more inputs than normal. A buffer is quite similar to a current amplifier as in case we are using a gain device to increase the signal current level. The amplifiers try to multiply the signal by a number. However the buffer is used to maintain the signal level at what ever level it is set. From the circuit diagrams below the current on the signal (V *in* ) has now only to supply a lower current to make sure that the given voltage is output.

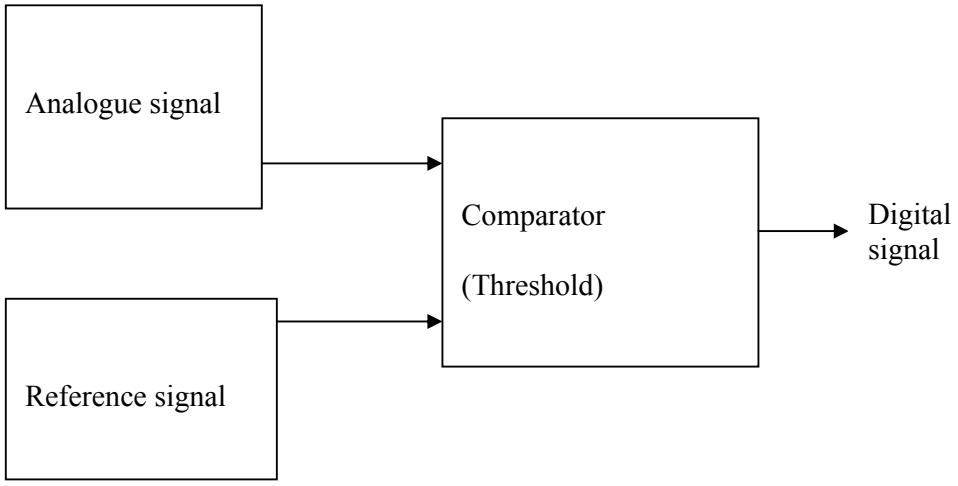



Transistor buffer



## Analogue to digital converters

The basic electronic system for converting an analogue signal to a digital signal is shown below.

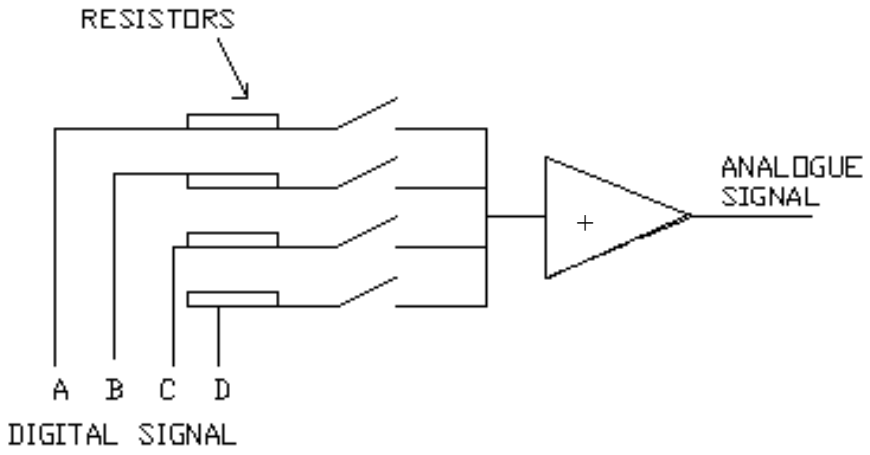



The environment around us is analogue. We convert these analogue signals to digital signals because the digital signals are much easier to deal with and perform action on. Electrical noise and distortion is not a factor in digital processing. For example a music CD converts analogue sound waves into digital and stores it in this format until it is played, when the CD is listened to the sound is analogue this is an example why we need converters to convert signals both ways.

Analogue to digit conversion is carried out by a technique called parallel encoding. The analogue signal is transferred to all the comparators parallel. The signal is then passed to the binary encoder to confer with the reference signal and produce an output.

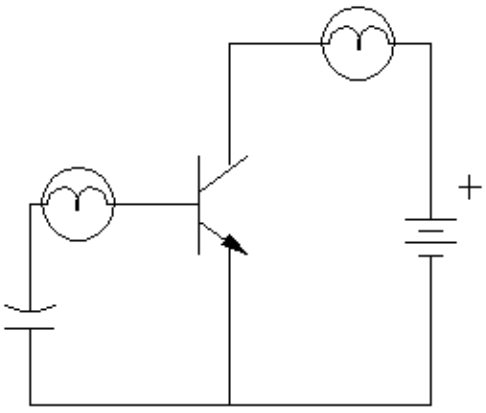
## Digital to analogue conversion

To convert digital to analogue signals weighted addition is used. Each bit of binary logic is given its appropriate analogue value by a summing amplifier; it is used to add the bits based on the value of a resistor connected to the inputs. The number of inputs can vary depending on the number of bits usually 4-8. Each of the inputs has a different resistor value and then these are added together to form an analogue out put. The diagram below shows a 4-bit digital to analogue converter.

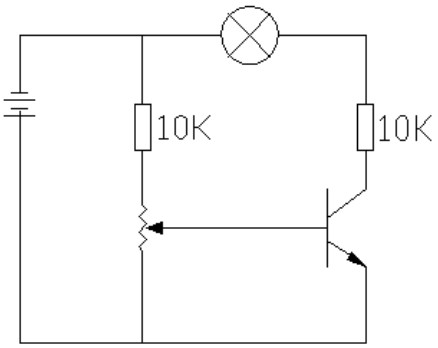


## Module 4: Task sheet

1. You are given two different transistors but you do not know what type are they. What test can you carry out to see which is which? PNP or NPN
2. In circuit diagrams how can you tell one transistor type from the other?
3. Draw a pnp transistor and index where the: collector, base and emitter.
4. The circuit diagram below shows a npn transistor in a circuit. Change this transistor with a pnp. Remember the npn has the positive of the supply connected to the collector and the negative to the emitter.

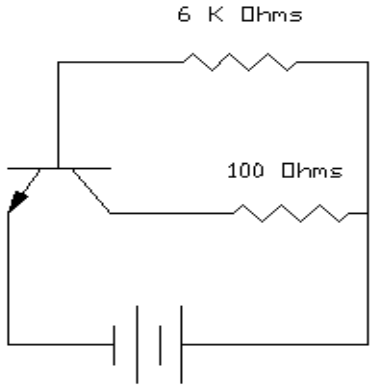



5. Explain what this circuit diagram represents.



6. Explain how an analogue signal can be converted into a digital signal and for what reasons might this be.
7. What is your understanding of dead band and bandwidth in relation to the Schmitt trigger?
8. How, using transistors carry out voltage amplification.
9. Find out from the Internet what is meant when a transistor is said to be saturated.
10. With the aid of a sketch showing its construction, explain why a small voltage change across the low resistance base emitter region of an npn transistor results in a much larger current change in the high resistance collector base region.
11. The emitter – base junction on a transistor is
    (a) Reversed biased to have high resistance
    (b) Forward biased to have no resistance.
    (c) Extremely thin
    (d) Composed of a p-type silicon
12. The arrow in the symbol used for a semiconductor diode points in the direction of
    (a) Easy electron movement
    (b) The + connection (positive)
    (c) Conventional current flow.
    (d) The connection to the supply

13. The current gain of the transistor in the circuit shown below is 100. Assuming that the base to emitter resistance is 0. (12 Volts)
    (a) How much (base-emitter) current flows in the transistor.
    (b) How much collector (collector-emitter) current flows in the transistor.




14. Design a circuit that uses two transistors. The circuit is to be used for a two player quiz game. When a person has the answer to the question they press the switch and a bulb behind their name lights up. If the second person presses the switch after the first person has the switch pushed their light will not illuminate.

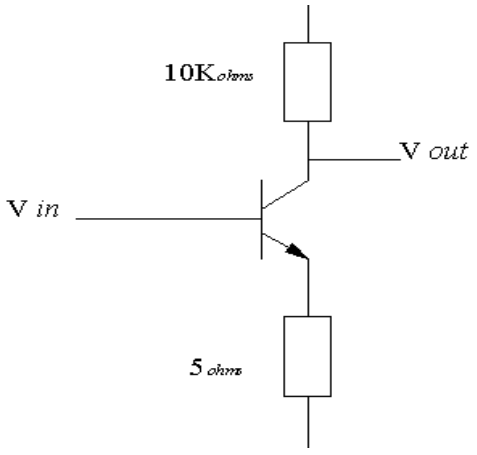
15. How is a double pole double throw switch wired to reverse the direction of a motor when flicked in one direction and then forward in the other direction.

16. Draw a systems diagram using automatic control, both open and closed loop control for a standard washing machine.

17. A toy that does not fall off the edge of raised surfaces or run into obstructions needs to be designed for a local company.
    (a) explain two different methods for sensing that an obstacle is present
    (b) draw a system diagram showing the main components of a toy control system
    (c) explain the function of each component and the relationships between them
    (d) how can a control problem such as sensing an edge at the same time sensing an obstacle.

18. With the aid of a diagram explain a digital signal and an analogue signal, what are the advantages and disadvantages of (a) transmission (b) storage of information by analogue and digital means.

19. Calculate the voltage Gain of the inverting amplifier shown below.



20. Can you give an example where a Schmitt trigger is used, and list the fundamental of this system?

21. A buffer is used to give a signal a “leg up” to a required level, what is the main difference with a buffer and an amplifier. (Diagrams will be necessary for explanation).

22. The comparator is used to compare two input signals hence the name comparator, but how can it be referred to as a threshold detector.

23. Using a systems approach explain how analogue to digital converters work and why such an operation is necessary.

24. If the out put of an amplifier for a guitar is 60 Watts and the input from the guitar is only 60mWatts, what is the power gain for that amplifier?

25. Would the power gain of the amplifier be greater if two guitars inputting a signal of 8mWatts were used instead? Give reasons for your answer.

## Module 5: Seven segment displays

Aim:

To demonstrate how seven segment displays operate.

Objectives:

At the end of this module pupils will be able to:

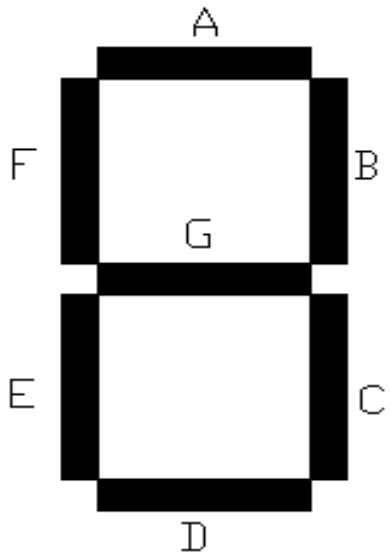
1. Conclude how seven segment displays work. (systems approach)
2. Construct a truth table for a seven segment display.
3. Display numbers in seven segments of display.
4. Test a seven segment display unit (old radio alarm clock, battery operated).
5. Formulate how a 4 bit binary counter operates the driver.
6. Explain why a decoder is necessary in this circuit.

Evaluation

Evaluation of this module may involve the pupils demonstrating how a seven segment display working and displaying preset and non consecutive numbers.

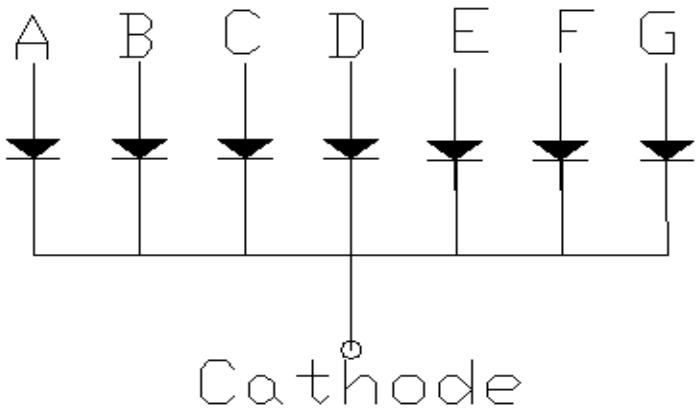
## Module 5: Seven segment displays

Seven segment displays are used everywhere form digital display alarm clocks to watches on cookers. The number 8 is selected as every other number can be made from this. LEDs are polarised and need to be connected the correct way round to operate. They use very small voltage and need a resistor in series with them. The seven segment displays use 7 LEDs in bar form to make up the number 8. To show numbers one to nine a decoder driver chip may be used and to reduce the wiring the display is either cathode or anode (sunk or sourced).



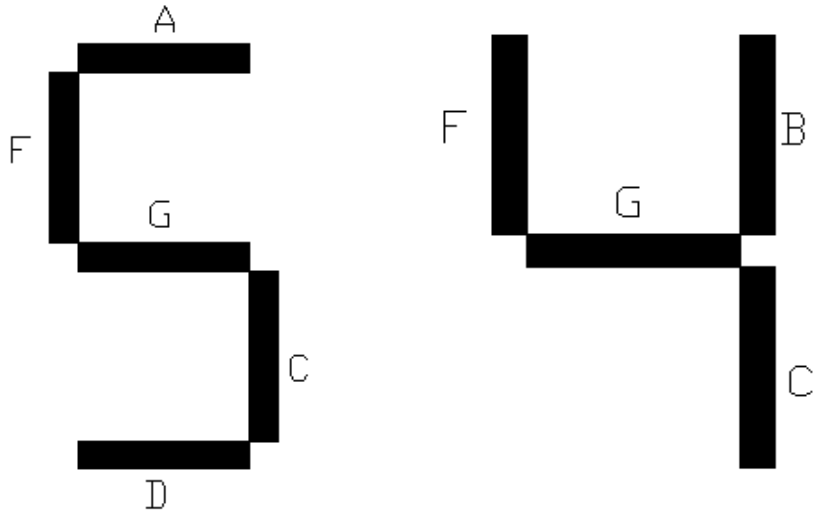

Some seven-segment display units may have commas and decimal points to make the reading clearer and some designs have the LEDs in italics. The LEDs displays have competition with liquid crystal displays and fluorescent. The liquid crystal displays are complex and are used in wristwatches and the fluorescent displays use high voltages. We will concentrate on the LED type display.

A circuit diagram for the above seven-segment display is shown below with the circuit sharing the cathode.




The seven-segment display can be purchased in different packages and of different size. Select a size that will cater for a demonstration on the mater breadboard. Depending on the set you have logic switches are usually present. Wire up all the logic switches and ensure that they are all turned to logic (0). Then turn on the power to the breadboard. Check each individual logic gate by turning it and seeing is the light working. Once the circuit is operational devise a logic table for each of the 10 numbers (0-9). The logic gates can be called after the segment that it displays. And the master breadboard can be used to test the theory.

| A | B | C | D | E | F | G | Output |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 9 |




Shown above are 5 and 4 and how they are represented on the LED display. From the truth table for the different (10) numbers logic statements should be written for them.

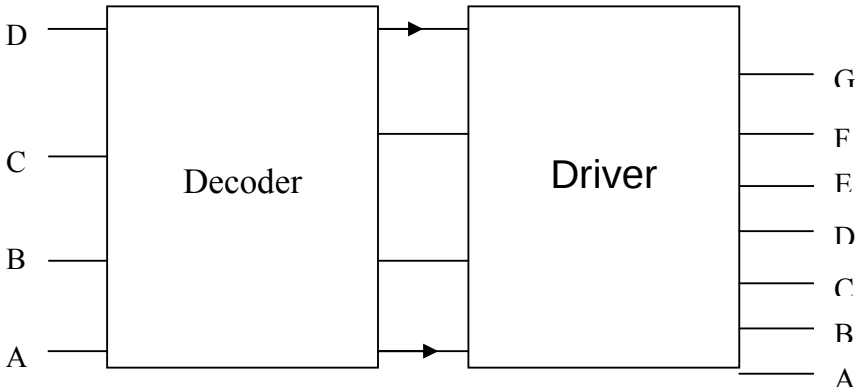
| | |
|---|---|
| 0= A.B.C.D.E.F | 0000 |
| 1=B.C | 0001 |
| 2= A.B.G.E.D | 0010 |
| 3=A.B.C.D.G | 0011 |
| 4=B.C.G.F | 0100 |
| 5= A.C.D.F.G | 0101 |
| 6= A.C.D.E.F.G | 0110 |
| 7= A.B.C | 0111 |
| 8=A.B.C.D.E.F.G | 1000 |
| 9= A.B.C.D.F.G | 1001 |

## Display drivers

From the above it is clear to see that we need a lot of **AND** gates to operate the display unit. This is not very economical therefore we use a decoder/driver. Decoding is necessary because the signal obtained from the Counter is not in the form that will run the display correctly. The counter is a 4 bit binary counter that will be discussed later. The decoder driver/receives an input signal from the counter this is then decoded and the driver sends out the output to the seven-segment display.

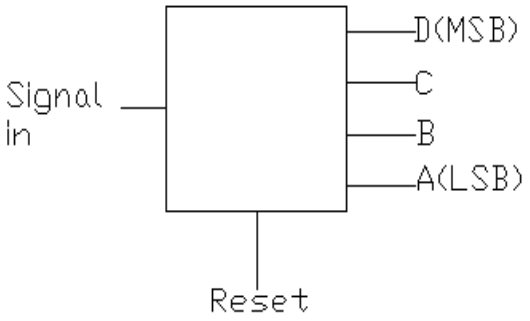
For example say we want to display 3 from the truth table and logic statement we want A.B.C.D.G to be at logic (1). The input from the counter can only be of four digits, 0010. These four numbers are decoded into turn A.B.C.D.G to logic (1). The driver sends out this signal and then the seven-segment display should light up 3.

If we wanted to display 4, we need B.C.G.F to be at logic (1). Therefore the 4 bit binary counter sends a signal 0100. This is decoded and the driver knows what this relates to, sends out the signal and 4 is displayed.

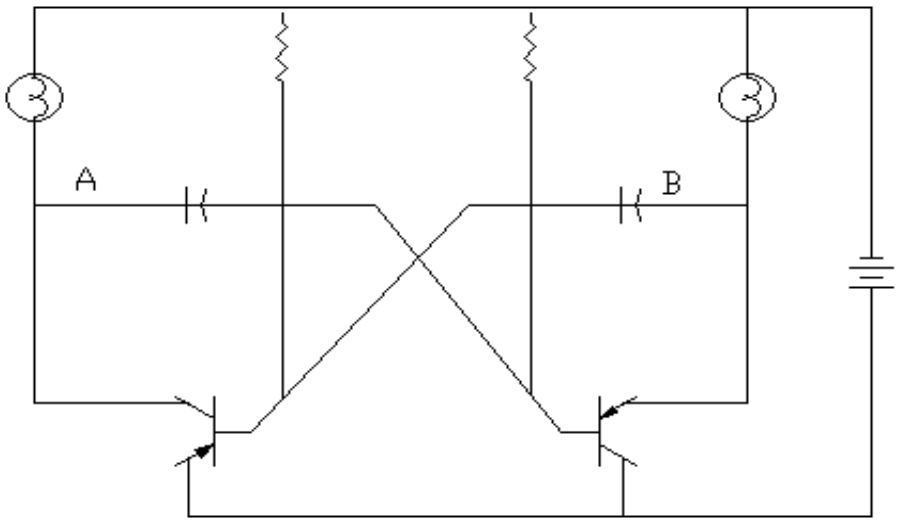


## Counters

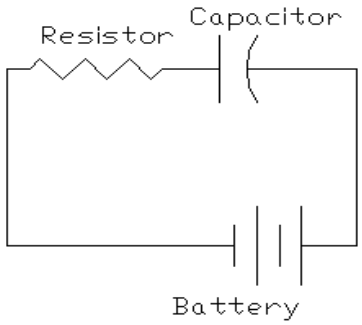
The 4-bit binary counter

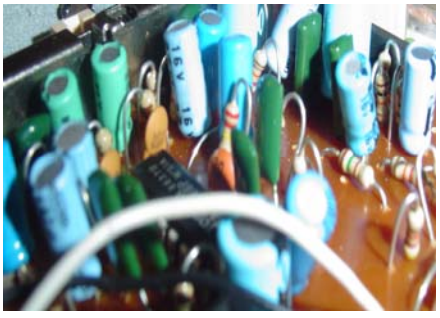



The 4-bit binary counter has one input and four outputs. It can be reset to 0000 when the reset signal is sent. Inside the counter is four flip-flops connected in series. Sequential logic is now used not just combinational. Sequential logic outputs are based on the state of the signal and the order in which input change. The output is also dependant on the timing of these input signals called pulses. Flip–flops are memory systems. Their make up is complex however they are just two circuits facing each other. In the diagram below the capacitor is connected to the collector of the opposite transistor. Its real name is an astable multibrator. The charging and discharging of the capacitors causes the switching motion in the A and B areas. Therefore the lights switch on and off in a flashing action, hence the name flip flop.



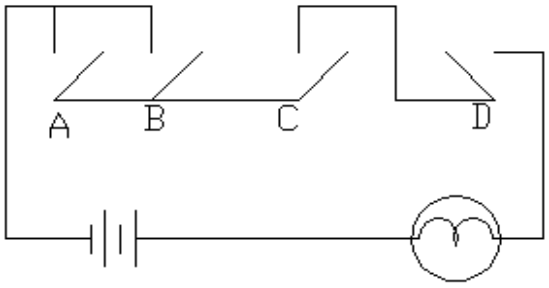
The 4-bit binary counter can be of two types, asynchronous counter or the synchronous counter. Asynchronous means that the timing or synchronisation of the pulse arriving at each flip-flop is not exact. This can lead to problems for example the timing of the segments on the seven-segment display. The synchronous counter has allotted for this and the pulse arrives at the clock input simultaneously. It is important to note that a 4-bit binary counter can only go between 0000 and 1111 this is 16 different changes. To reset the counter after it has counted from 0- 9 is carried out by a **NAND** gate. The outputs from terminals are **NAND**ed together and feed into the reset input and the counter starts again. In the case of the seven-segment display will be reset after every 10 times. This memory in binary digit form can be transferred one bit at a time or many bits at a time. The one bit process can be slow and tedious and is called serial while the several bits at a time are called parallel. This data is stored temporarily in a device called a register. Registers can multiply or divide data also. Circuits based on real time use capacitors. Knowledge of capacitors is expected. When experimenting with a timing circuit use a resistor of high value to delay the charging process. The time constant with is the time taken to rise to 2/3 of the battery voltage is = capacitance X resistance. However some capacitors have large tolerances. Capacitors are fundamental in the tuning of radios.



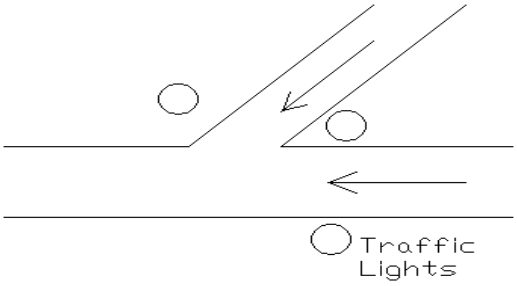



## Module 5: Task Sheet

1. What do MSB and LSB on the 4-bit binary counter mean?
2. What is the function of the decoder in the counter?
3. Explain the difference between an analogue signal and a digital signal.
4. Pulses are referred to when talking about 4-bit counters. What are they
5. This data can be sent to each flip-flop simultaneously what is this counter called.
6. With the aid of truth tables explain the **NOT** gate.
7. What is the primary reason why a cathode or anode display is used?
8. Draw a truth table for the circuit below and draw a logic circuit as your solution to the problem.




9. The diagram below shows a street intersection. Design a control system that will operate the traffic lights. The car arriving at the intersection first must get the green light, but if both cars arrive at the same time give priority to the car travelling in the straightest direction.



## Module 6: Open and closed loop control

Aim:

To introduce the concept of feedback in electronic systems integrating servos and stepper motors.

Objectives:

At the end of this module pupils will be able to:

1. Contrast open and closed loop systems.
2. List possible advantages and disadvantages of both systems.
3. Explain and describe the principles and operation of a servo and a stepper motor.
4. Critically analyse each possible loop system with its relevant components to determine their suitability to a given application.

Evaluation

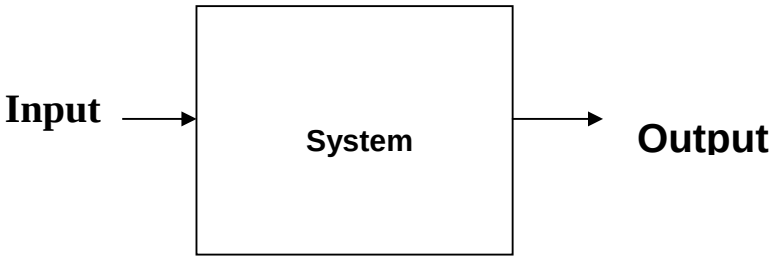
To evaluate this module an experimental approach can be taken, servos and stepper motors are project components and pupils should be grounded in the basics that will lead then to greater independent learning in the class room setting. The task sheets can test pupil's theoretical knowledge but practical hands on should be encouraged.

## Module 6: Open and Closed loop control

The control aspect of robotics deals with the brain. This part directs the manipulator what to do, when to do it, and how to do it. There are two types of control methods and they are open and closed control.

### Open loop Control

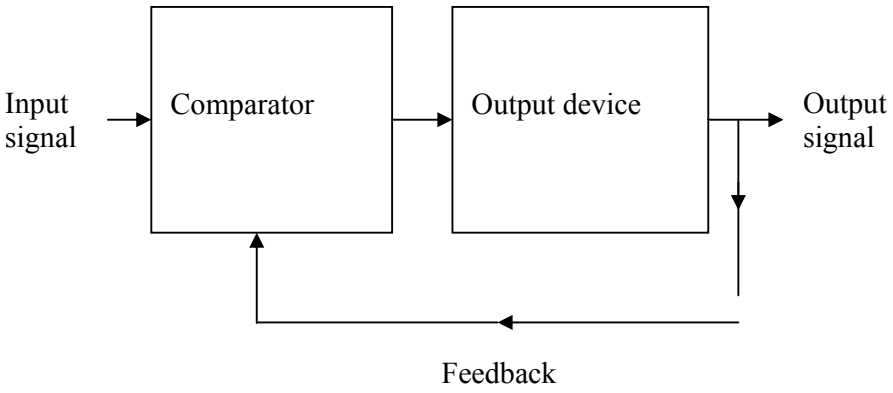
As mentioned earlier some systems use feedback as a means of communicating with the outside environment. Open control systems operate with out feedback. They are also called non servo-controlled systems. On/off switches usually control robots that operate using this system. The light switch in a room is an example of an open-loop system.




The open loop control takes no action to see is the output correct. Another example of an open loop system is a garden sprinkler worked on a timer. The sprinkler turns on every day for 1 hour. This is effective during hot weather spells but when it is raining if the sprinkler is not turned off the sprinkler will still sprinkle water for one hour a day.

## Closed loop control

Closed loop control uses feedback as a means of controlling the robot. Remember feedback is the information received regards where a motion is and how it is performing. The output is compared with the input so an appropriate action of control can be taken. The robots responses are measured and sent back to the control system and this control system where they movement/motion of the robot is altered in accordance with the feedback information. A typical example of the light bulb using feedback is that during the day (bright) the light bulb might be on at ¼ power and during the night (dark) the bulb is on full power. The garden sprinkler may use a moisture detector and only turn on when this is below a preset percentage and avoid turning it on when it is raining. Closed loop control is referred to as servo-controlled systems. Humans operate a closed loop control system. A fully automatic system is shown in the diagram below.




There are two types of feedback, negative and positive. Negative feedback subtracts form the input signal; this directs the output towards a set level. This negative feedback allows for precise control of a system. Positive feedback adds

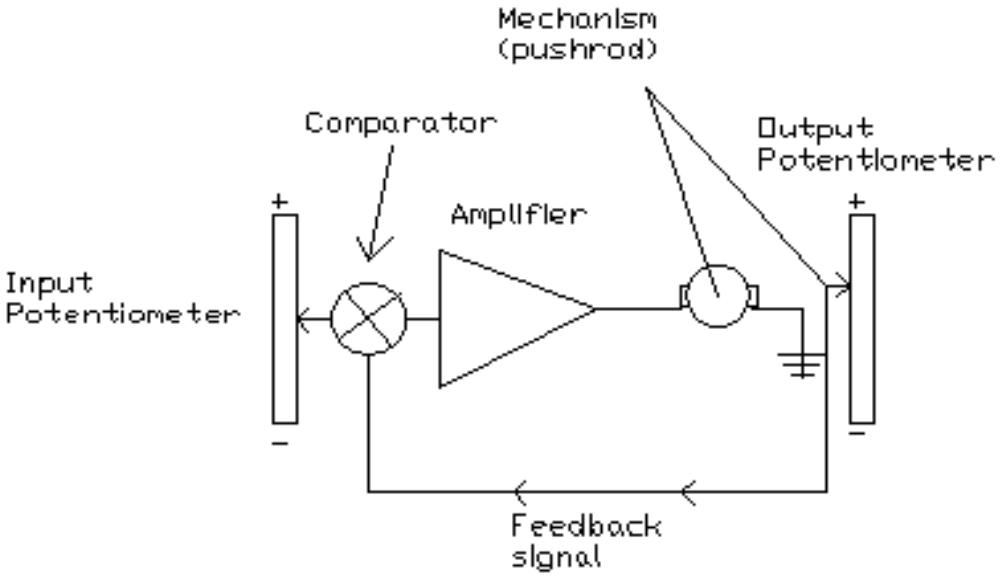
to the input signal and directs the output away from its existing state. This leads to an unstable output. Closed loop systems require sensors to provide them with feedback information. This information is sent in the form of a signal. To process these signal the robots must have a comparer to compare the signal with, an amplifier, a sensor to create the feedback and the feedback signal. If the input and feedback signals are different, the comparator will transmit some type of difference signal causing the device output to act differently.

## Operation and control of dc servos and stepper motors

A direct current servo is a motor attached to a feedback device. There is an output shaft that can be positioned to specific angular positions by sending the servo an input signal. The servo is used in remote controlled aeroplanes to power the throttle. Servos are very expensive machinery and are only used when they are really needed. The servo has three electronic wires exiting it and a plate fixed on to the rotation shaft called a control horn where push rods are connected. As long as the signal exists on the input line the servo will not rotate, but when the signal changes the angular position of the control horn changes. If a strain from an environmental source acts on the control horn the servo will resist this and try and hold its signalled position. Servos are very small and powerful they can be easily fitted into robots. They also come with the feed back mechanism and electronics completed.

## How do servos work

Servos have a motor, control circuit and a potentiometer. The feedback device is the potentiometer. The motor turns the potentiometer as well as the shaft. The result signal from the potentiometer is then fed into the circuit and the motor keeps turning the shaft until the desired signal is achieved. Depending on the type of servo the rotational angular degree limit is 180-210 degrees and reverse. The principles of an analogue servo are shown on the next page including a feedback mechanism.




A signal is sent to the input potentiometer to tell the device what angle the motor should be at. A signal from the out put potentiometer tells the comparator where the motor actually is. The comparator finds the difference in the two signals and this difference is amplified to produce a current. When the motor is where the input signal requires it to be the two potentiometers will have the same reading and the comparator will read no difference so the motor is in the correct position. The three electrical wires entering a servo are terminals positive, ground and signal wire. The duration of the pulse determines the angular motion in the servo, for example the duration for a pulse may be 2 mille seconds and this moves the input potentiometer to equal this then the motor will turn 5 degrees. All of this information is supplied in the packaging of the servo. Servos can come in analogue or digital form. The analogue signal is received at 30 times a second while the digital signal can be received up to 280 times a second. This

difference is important for reaction times on robots especially their grippers. The rate at which a signal is received is called its frequency.

### Stepper motor

A stepper motors rotation motion is in discrete steps. Unlike the normal motor whose motion is continuous. Direct current motors emit high levels of electrical noise and this can be herd by the human ear. The transferring of current from one sector of coil to another as the conductors/bushes rotate causes this noise. When a stepper motor receives a signal in electrical pulse form it rotates the drive shaft a certain amount. The speed and direction of the shaft is dependent on these signals. Each time a pulse is recorded the motor rotates for a known amount; this amount is dependant on the motor and the length of signal. The usual rotation for a signal pulse is 7.2 degrees and if 50 of these pulses are in quick succession the shaft will complete one full rotation. Stepper motors are usually purchased with pre-fitted gearboxes to increase torque.

### How does a stepper motor work?

Stepper motors have either 4 or 6 wires, due to the increased number of coils. This increased number of coils allows the shaft to step rotate. When these coils get a signal or pulse this will cause the shaft to rotate until the coils are in line with the bushes. The direction of rotation is achieved from the sequence for these pulses. The rotation speed is determined by the frequency of these signals, audio signals can make stepper act like an ordinary motor. The disk drive, robotic arm and end-of-arm tooling use stepper motors.

## Servo or stepper motor

In robotics the decision of opting for a servo or a stepper motor often arises. They both carry out the same task but at different levels. Stepper motors do not require a feedback mechanism where a servomotor needs this feedback mechanism to function properly.

In robotic applications repeatability and positioning are critical, with a servo the system is a closed loop and can relate with its environment and change with its changes.

For example a bolt might be out of alignment with an engine block the servo can feedback this information a produce a corrective action and deal with it. This is not the case with a stepper motor, if the stepper motor is out of sink with the rest of the operation due to over loading it does not have the ability to correct itself. Servomotors are capable of high rotational speed and they have a great ability to remember where they are.

Stepper motors lose their positions quite easily when high accelerations are needed (screwing in a bolt).

Due to their increased level of technology servomotors are more expensive than the stepper motor.

Stepper motors are now being supplied with integrated circuits and can be controlled easily with computers due to the fact that they have between 4-8 wires leaving them. The SAA1027 stepper motor driver is one such example.

## Principals of combinational and sequential logic

Combinational logic circuitry depends on the number of inputs and the combination of the logic gates. There is no feedback loop in combinational logic. Truth tables and logic statements can define a combinational logic circuit.

Sequential logic circuitry outputs are dependent on the inputs and the sequence of inputs. The main difference with both types of logic is that the sequential logic has the ability to store memory and has feedback loops involved.

## Task sheet Open/closed control

1. What is the difference of a system having an open loop control and the same system having a closed loop control system?

2. List the advantages and disadvantages of such systems (two different examples are required).

3. The term feed back can be associated with either positive or negative type. Which type is used for precise control? What is the use for the other type of feedback?

4. Explain the principles of a stepper motor?

5. How do servos that are radio controlled operate? A circuit diagram is required and state the function of the potentiometer in the circuit?

6. Logic systems require gates, what is the difference of gates in sequential logic and gates in combinational logic?

7. What criterion is used when selecting a servo instead of a stepper motor for a product?

8. Draw a cross section diagram of a stepper motor naming at least 5 parts, the diagram should be referred to when explaining its operation?

## Module 6: Robotics

Aim:

To introduce robotics, classify robot joints and classify robot structure.

Objectives:

At the end of this module pupils will be able to:

1. Demonstrate the term “degrees of freedom” and illustrate the six lower pairs.
2. Analyse robot structures (5) in relation to degrees of freedom and co-ordinate frames.
3. Calculate forces and moments on robot links.
4. Evaluate the different power sources and end-of-arm tooling

Evaluation

This module will be particularity hard to evaluate. Pupils should be encouraged to use Lego to demonstrate the degrees of freedom. The robot structures may also be constructed using Lego and using practical examples of such structures.

## Module 6: Robotics

Robotics is the study of robots. There are many different definitions for the term robot and the most common meaning is that it is an automated machine, performing functions in a human manner. The American Robotics Institute defines a robot as:

> "A reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices, through variable programmed motions for the performance of a variety of tasks"

A Czechoslovakian artist Karel Capek first used the term robot in a play that he wrote, which he had robots staring. To be called robots they must have is a power source, controlling device or brain (computer) and an output that can be made perform tasks. Robot usage is becoming wide spread for research robots to domestic house cleaning robots. Robots out weight automated machines as they can be reprogrammed to carry out different operations. The robot is versatile and creates motion, can be used in dangerous situations and is never late for work. Robots fit into many different categories and are not just industrial. Toy robots such as the Furby were a major success and military robots are becoming more common. Robots out perform humans in many different aspects, for example pick up a pen and place it on the floor then lift up and repeat this 1000 times. The human body and mind will not fancy doing this while a robot can do it forever. Robots can be stronger and lift up to 10 times their own body weight. Robots however are not designed solely

for strength they can perform very delicate procedures when precision is needed.

## Robot history

The first device to use feedback was designed by James Watt. After designing and modifying his steam engine Watt wanted to control the speed of the shafts. He devised Watt's fly ball governor. This devises connected to a value limited the amount of steam that passed it. When the steam was too high and the shafts were rotating fast and the balls of steel would move out and partially close the value to regulate the speed. Our bodies use feedback.

Programming was first used in the automatic looming industry. Cards with holes in specific areas were used and told the loom when and where to change the tread. Electricity discovery in the late $19^{th}$ century was a major step in the world of robots. The invention of the first computer was in 1939, these were large and were not good as they were continuously breaking down. The invention of the transistor solved most of these problems. It took until 1955 until the first robot was used in industry called the Unimate.

Today the transistors are on silicon and millions are placed inside small chips. The CPU of most computers is on single chips called microprocessors. Inside these processors are circuits arranged into the famous logic gates. Lets discuss robots/robotics and there design, make up and robotic control.

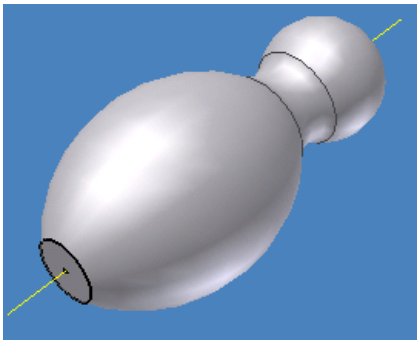
## Classification of robotic joints

The degree of freedom and co-ordinate frames classify robotic joints. The robot anatomy is made up by rigid links jointed together. These links can be jointed together in series to form an open loop or they can be jointed in a complex manner to form a closed loop. We can consider the links to be rigid and have the ability to perform the task in hand. They also are capable of six degrees of freedom. Now we need to join these links.

## Joints

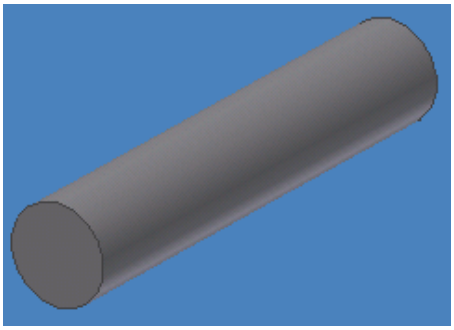
There are limitless ways of jointing links together from a simple hinge to a complex ball and socket joint. A German mechanical engineer Franz Reuleaux defined a joint system called lower pairs. The Reuleaux lower pair is a combination of six joints, with each containing one solid and one hollow. The surface of the solid is in contact with the hollow constant and allows relative motion.

1. Any surface of revolution gives a revolute pair or R-pair. This is a one-degree of freedom joint as it only allows revolutions about the fixed axis.

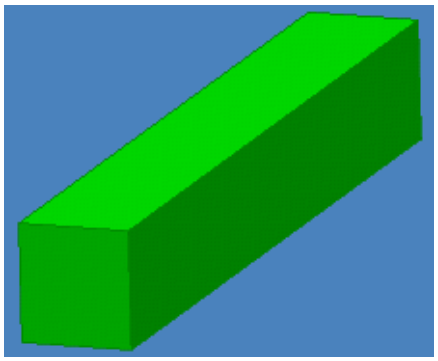


2. Any helicoidal surface gives a screw or H-pair. An example of this is a nut and bolt. This joint is still referred to as a one-degree of freedom joint even though the bolt moves up and down. We still only need one parameter to give us position.

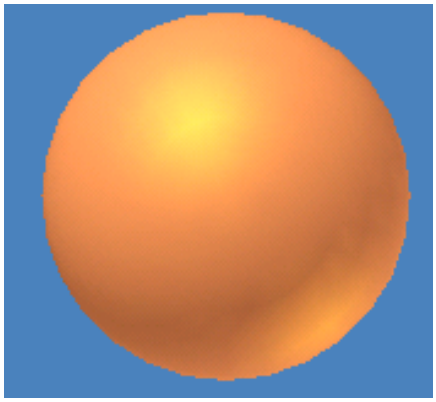
3. A surface of a cylinder that is one of both translation and rotation is known a cylindrical or C-pair. This cylindric joint has two-degrees of freedom as two parameters are needed due to rotations on its axis and translations (sliding movement). A typical example of this motion is extendable vacuum cleaner hoses.



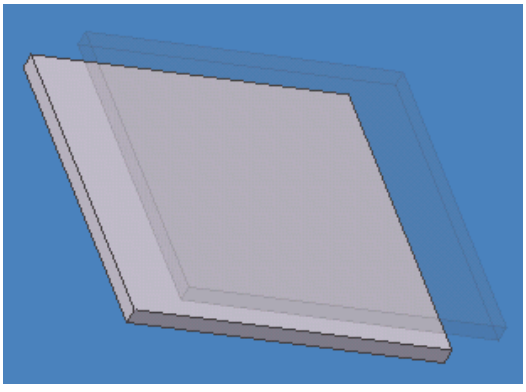
4. The surface of translation alone. (Prism) this is called a prismatic or P-pair. This type of joint has only one-degree of freedom.



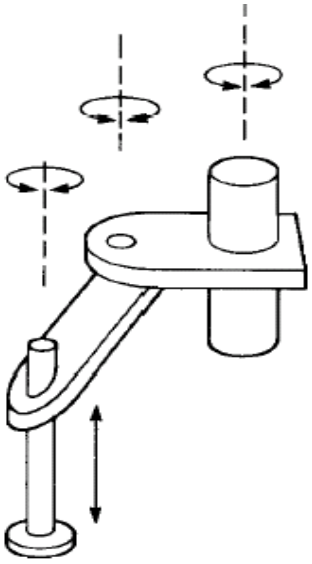
5. A sphere is a surface of revolution about a diameter. The ball and socket are an example of such a Pair. This is called a spherical pair or an S-pair. The spherical joint has three-degrees of freedom and we need three parameters to specify the move.



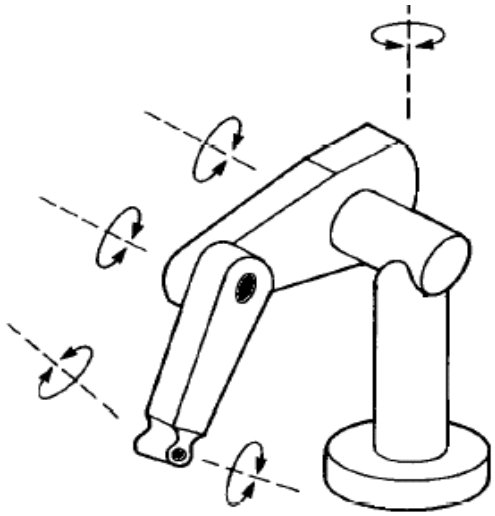
6. A plane is a surface defined by three points. It is also a translation of any given line and a surface of revolution about any line. Two planes form a planer or E-plane. In this type of joint only planer movement is permitted. Three parameters are required to specify such a movement; they are the lengths of translation along two directions and the angle of rotation about a fixed point.



Robots can have different degrees of freedom depending on what they are designed for. Assembly line robots usually only have 4 degrees of freedom. The first 3-degrees are required to orientate the object and then final degree is to move it into a different plane (lifting).



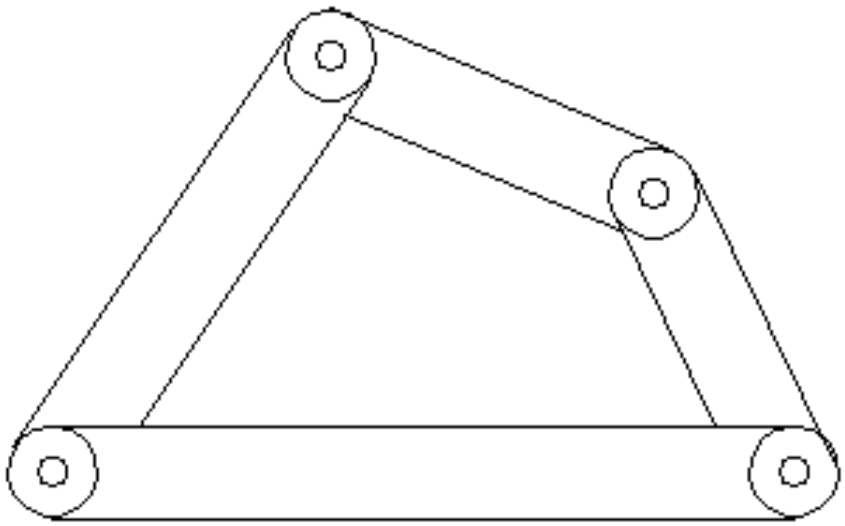
Robots with 5-degrees of freedom are used in applications such as arc welding and spraying. The orientation of the tool around the final axis is not needed as the welding gun can be held at any direction to the work piece.



We have classified robotic joints by there degrees of freedom and now we will concentrate on the co-ordinate frames.

## Co-ordinate frames

Co-ordinate frames are the most common method that the robot uses to keep track of the links positions. We can always use 3 points to define a co-ordinate frame (vector geometry). Vector geometry for some part is easy and it can define where a link is in space. However if working with a robot that has 3 links Co-ordinate frames can become quite complex and a different method is needed. The X, Y and Z co-ordinates used here can be easily displayed by changing the UCS in AutoCAD. The most common co-ordinate frame is a four bar chain mechanism this is an open loop system (shown below). With four bar chains movement is possible and each pin joint has a distant pattern. This pattern is called a locus. Four bars pin jointed frame.



## Forces and moments

Newton's second law of motion defines a force.

Force = (Mass) (Acceleration)

The mass is in kilograms and the acceleration is in metres per second per second. The unit for force unit is Newton's.

Weight and mass are not the same quantity. The weight is equal to the mass of the object multiplied by the acceleration of gravity.

Weight (N) = (mass) (9.8 m/s^2)

Say that a car is travelling at an acceleration of 30 m/s^2 and the cars mass is 700 Kg. What force is the car exerting?

F = (Mass) (Acceleration) 21000=30 X 700 so the car has a force of 21 KN.

Another example is to find the weight of an object whose mass is 10 Kg, and its acceleration when a force of 200 N acts on it.

W= (mass) X (gravity acceleration)

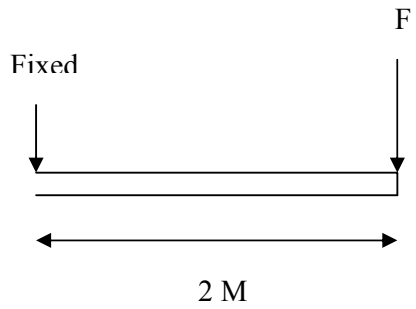
10Kg X 9.8 m/s^2 = 98N

Force = Mass X Acceleration Acceleration = Force/Mass

? = 200N / 10Kg Acceleration = 20m/s^2

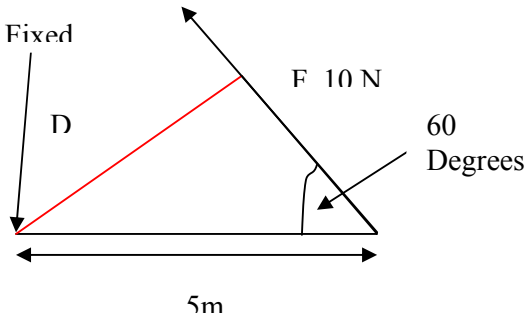
A moment of a force can be defined as a point at its magnitude multiplied by the perpendicular distance of the point from its line of action. The moment of force is the Newton metre (Nm). This definition becomes clearer after some examples.

For example calculate the moment of force acting on this robot arm. The force acting down is F 100N and the length of the arm (link) is 2 metres.




The moment of force = 100N X 2m = 200Nm

Forces can be applied at angles to the robot arm and trigonometry is used to find the moment of force. The diagram below shows a perfect example of a force acting at a known angle.




The moment of force around the fixed point has to be perpendicular from the force to the fixed point. This is drawn in red D. The length of this line is now required. D = 5 sin 60 degrees= 4.33. The length of D we now have and to calculate the moment of force we multiply this length by the force. =43.3Nm.

## Classification of robots based on structure and application

Robots have lot of different structures the copy of the human arm is most common. The base of the robot supports the arm and is called the shoulder. The shoulder allows for rotation. The arm is the main piece of the robot. It pivots on the elbow and on the shoulder joint. The wrist is the cross hairs for the hand. The wrist has many motions and is dependent on the design. The rotary motion of the wrist is called the roll, over and back motion is yaw and the up and motion is the pitch. The robots hand is called the gripper.

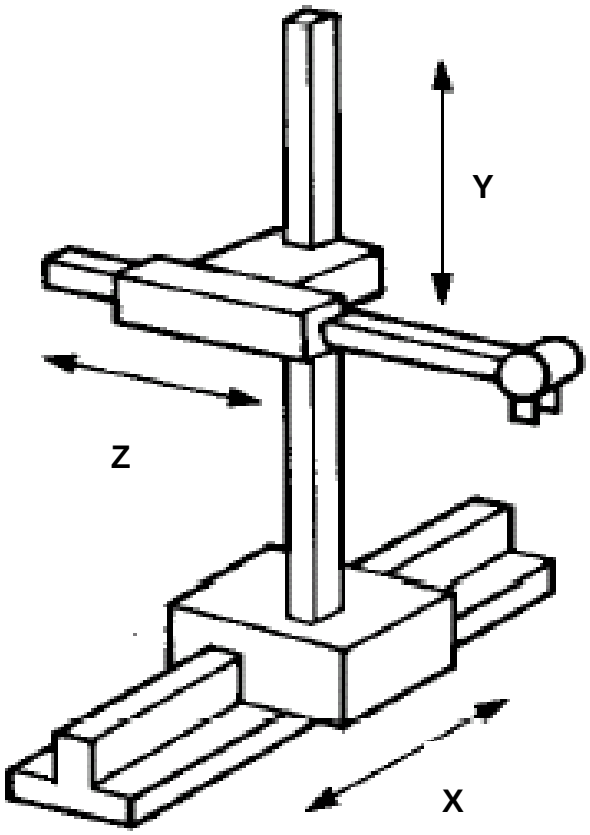
Robot structure can be broken down into 3 parts; the first part is the end-of-arm tooling this is that part that actually performs the task. Secondly what power source is the robot using? And thirdly the manipulator the part of the robot that is concerned with motion for example the arm.

We are concerned with the manipulator (industrial type robots) and their structure and application. Industrial type robots come in many forms; we will discuss each of these forms and note where they are used. The five types are Cartesian coordinates, SCARA, cylindrical coordinates, Polar coordinates and jointed arm robots.

Typical application of industrial robots, Casing, forging, assembly, arc and spot welding, spraying, plastic moulding and machining like drilling holes to fitting.

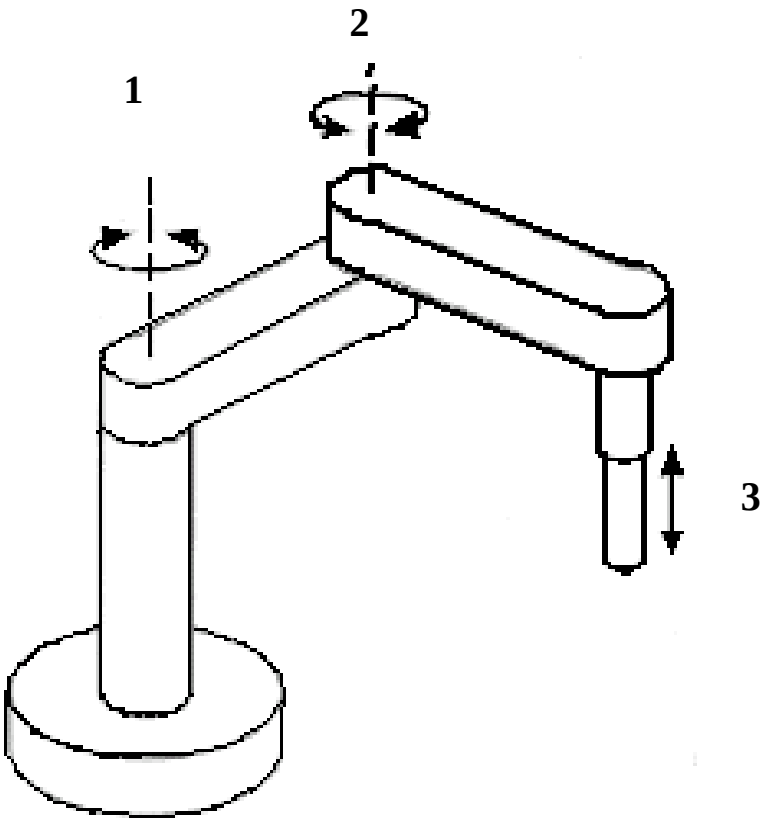
## 1. Cartesian coordinates

A Cartesian coordinate's robot moves in straight lines, this motion is along three axes: over and back, in and out and up and down. Other names for this robot are X-Y-Z robot and rectilinear robot. The AutoCAD plotter uses the same degrees of freedom as this robot.



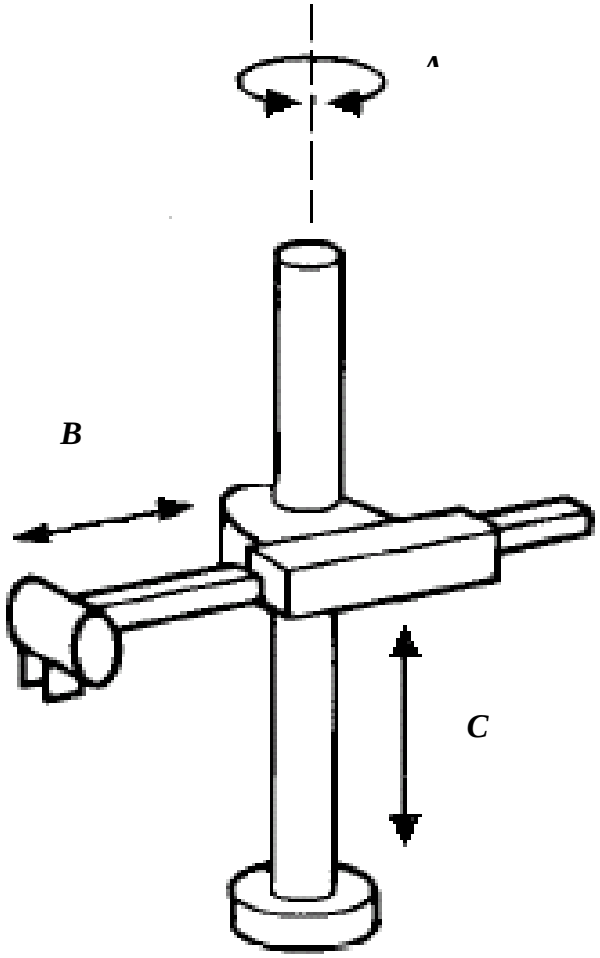
## 2. SCARA (Selective Compliance Assembly Robot Arm)

The SCARA robot has its shoulder and elbow rotational axis in the vertical plane. This makes the arm very strong in the vertical direction but it is not very strong in the horizontal plane. The reach axis has a rotational joint in the plane parallel to the floor. Adjustable lighting in garages and film sets use this type of structure.



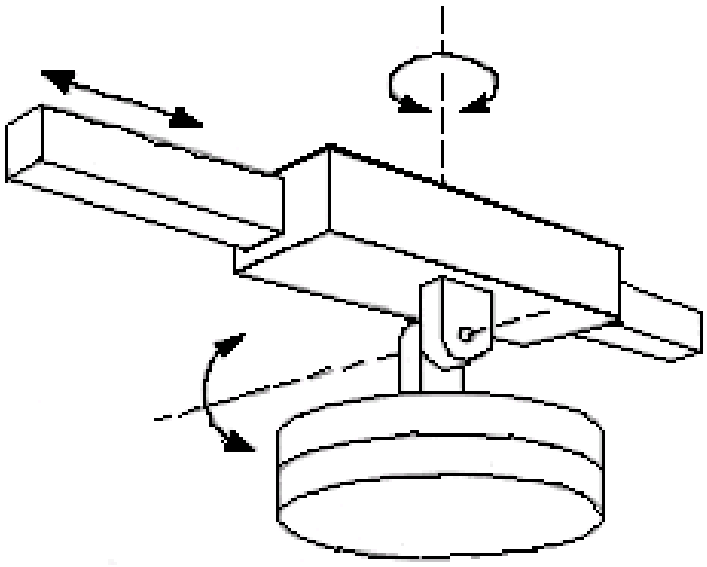
### 3. Cylindrical coordinates.

The configuration of this robot consists of a vertical column sitting on a base that lets the arm move up and down. The arm can be adjusted in and out in line with the cylindrical column. A building crane has the same features as this robot. This robot displays 3-degrees of freedom.



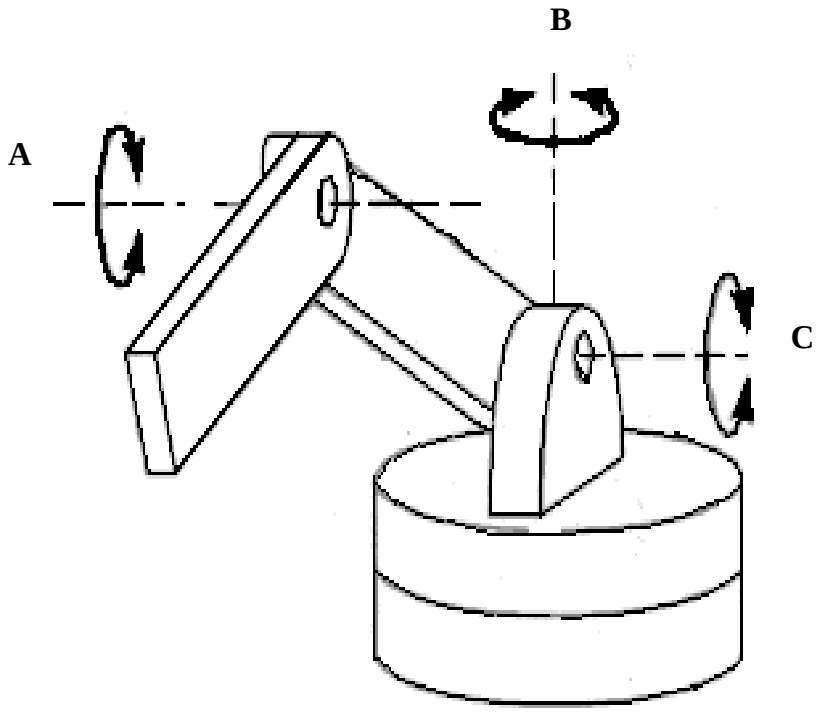
### 4. Polar coordinates

This robot is often referred to as the spherical robot. It allows for rotation about two different axes. The horizontal base has an axis to position the robot and the vertical axis is used for lifting and lowering. Attached to this axis is a P-pair joint that allows the arm to move in and out. Military robots that launch rockets use this type of robot.



## 5. Jointed arm robot

This robot arm has the general configuration of the human hand. This arm is also called the revolute co-ordinates robot. It has the possibility of rotating around the base in the horizontal plane. The vertical plane contains two more rotational joints. The robot has a base/column, shoulder and elbow. This robot has 3-degrees of freedom. A perfect example of this is the excavation digger or Hi-Mac.




You can see the limits and advantages one robot configuration has over the other. As robots are most commonly found in industry each of the above robots

have a manufacturing application. However we are missing parts of the robot. We have dealt with the manipulator and now need to discuss the power source and the end-of-arm tooling.

## Power Source

The main power sources for robots are electrical, pneumatic and hydraulic.

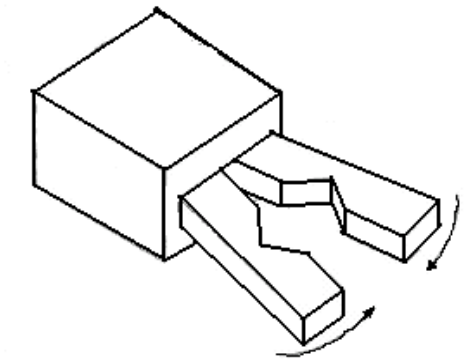
Electrical power source for robots is most common. Robots prefer direct current due to the electronics systems that operate them. The robot either has it own alternating current to direct current converter. Robots in industry will run on a system of power supplies and only portable robots for example bomb diffusion robots will have a battery pack.

Pneumatic power sources need a compressor and storage tank. The power is the air pressure that the valves realise. The pressurized actuating cylinder is the driving tool used. Air pressure is reliable and can be measured using automated systems. Different cylinders can be used the two most common is the single and double action cylinders.

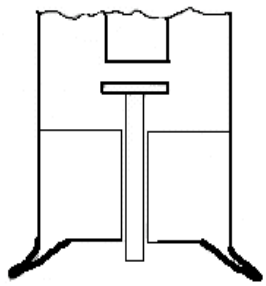
Hydraulic power uses oil compressed in rams. Oil pumps are used to create the pressure and move the pistons up and down the cylinders. Very high pressures can be produced. Pneumatic rams can be spongy but the oil forms like a solid in the ram when the pressure is increased.

## End-of-arm tooling

End-of-arm tooling may add between one-three more axes of motion to the manipulator. The hand is often called the gripper or actuator. There are many different forms of grippers and they usually use the same power source as the manipulator apart from some robots. Many use springs to close the gripper quickly. Grippers may not always be used for example a spraying robot may not require a gripper. The gripper is always purchased separate from the manipulator. Some robots even change their gripper during a task. Remember robots can be used to do tasks that humans cannot do, lift red-hot steel and toxic materials, and tasks that humans can do, perform drilling and riveting. Research and development into the gripper has not reached the same effects as the human hand just yet. The gripper design can very from using suction to using electromagnets. It must be notes that grippers are specially designed to grip the object that it is required to grip using it in another application can work but usually is unsuccessful.



**Gripper**

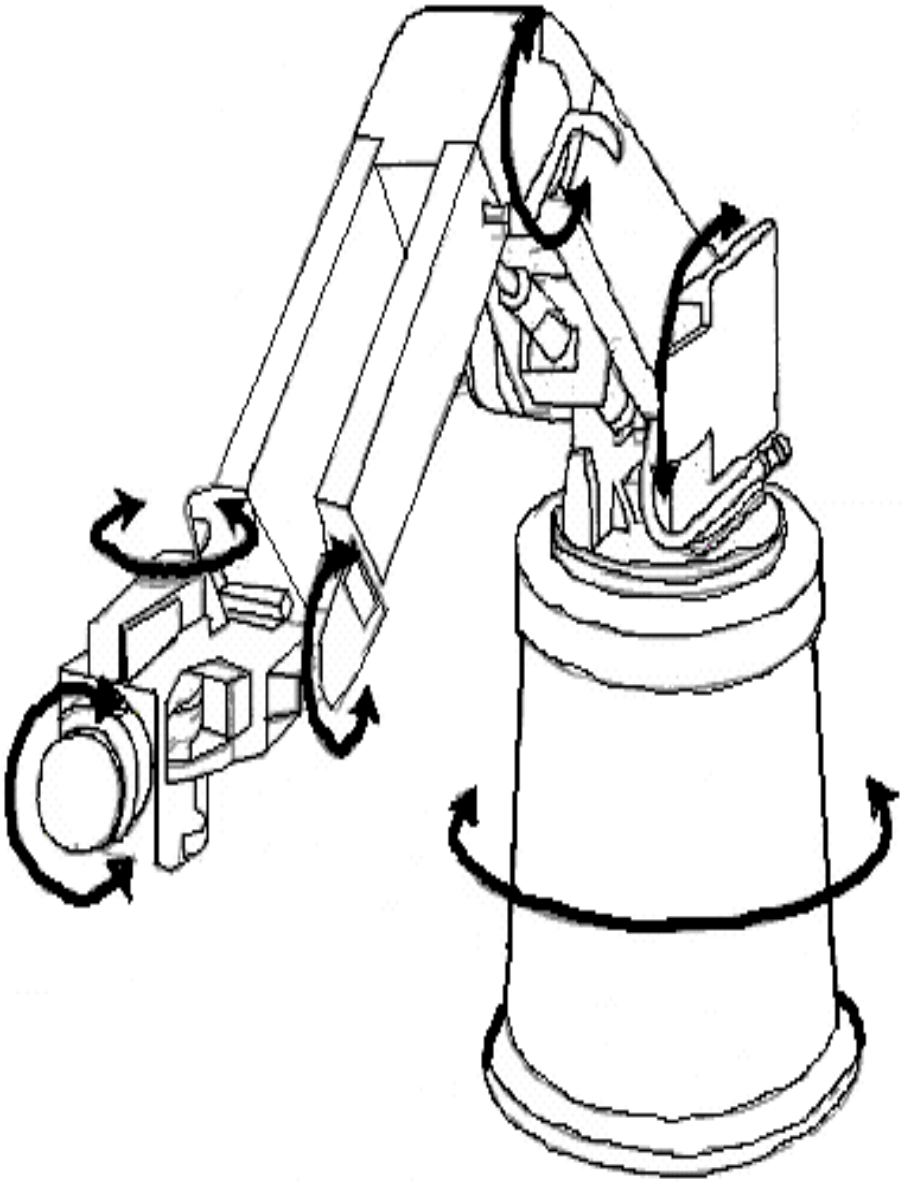


**Suction grip**

Robot with six degrees of freedom

3-degrees on manipulator

3-degrees on the gripper



## Locomotion

Robots in factories tend to stay in the same place and the work around them tends to move. This is particularly true in car manufacture factories and on other production lines. The area in which robots work within is called their work envelope and this is usually measured by the swing displacement X maximum displacement in the vertical plane.

Some robots are required to move around and wheels, tracks or even legs carry out this locomotion.

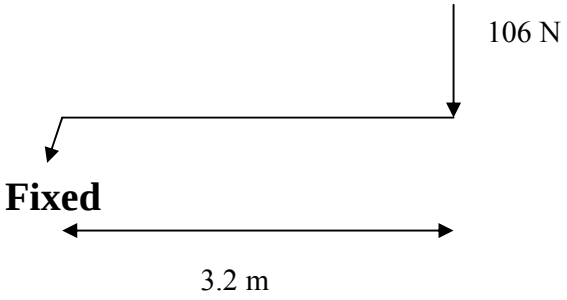
Wheels are the most common method of robot locomotion and they are the fastest of the three types. Wheels however are not good at crossing uneven ground.

Tracks are quite slow and require high levels of power. They are good at crossing both even and uneven surfaces.

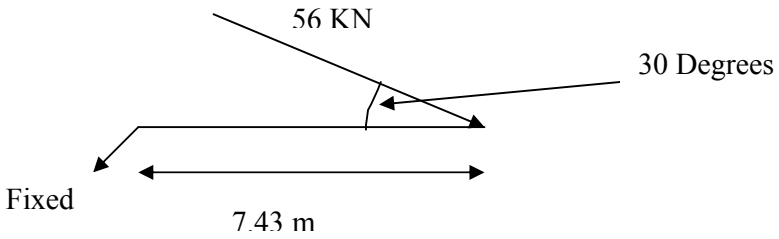
Legs are great for climbing objects; they can climb upside down with suction pads and can scale poor surfaces. They are very hard to design and quite difficult to build.

## Task sheet

1. Define what a robot is in your own words.
2. Describe your arm in relation to degrees of freedom and relate this to robots
3. What is a prismatic joint?
4. A revolute joint displays the same characteristics as a spherical joint, however the spherical joint has more degrees of freedom. Explain why with the use of diagrams.
5. Calculate the weight of a robotic arm whose mass is 175Kg and find its acceleration when 700N is acting on it.
6. Calculate the mass of a gripper whose weight is 500N.
7. The moment of force acting on the robotic arm cannot be greater than 340 Nm for safety reasons. Is the situation below safe?




8. What is the moment of force acting on this robotic arm link.



9. How are robots classified?
10. The cylindrical co-ordinates robot has three degrees of freedom and the SCARA robot has this amount also. Contrast both robot designs from diagrams and include the joints as a fundamental part.
11. Describe the term manipulator and explain what other part is needed for the robot to function properly.
12. Robots use different power sources why is this variation needed.
13. End–of–arm tooling can make or break a robot. Explain what this state means.

14. Design a robot manipulator that can lift vertically in a straight line and rotate the lifted object through 90 degrees in the horizontal plane. Your design should have at least two-degrees of freedom and no more than four-degrees of freedom.

15. Design a gripper that can pick up 5mm pan head bolts and screw them into a pre-tapped metal base. Sketches and idea.

16. Robots are finding usage in common day work places, research one such place and find out how this has benefited or hindered the people that work with such a robot. (Internet)

## Module 7: Flow charts

Aim:

To further develop pupil's problem solving skills by assisting problem understanding and a logical pattern to solving.

Objectives:

At the end of this module pupils will be able to:

1. Construct flow charts using Microsoft Word.
2. Analyse and break down problems into manageable steps.
3. Work in groups and teams to test their flow charts for clarity and practical application.
4. Associate the flow charts with computer programming.

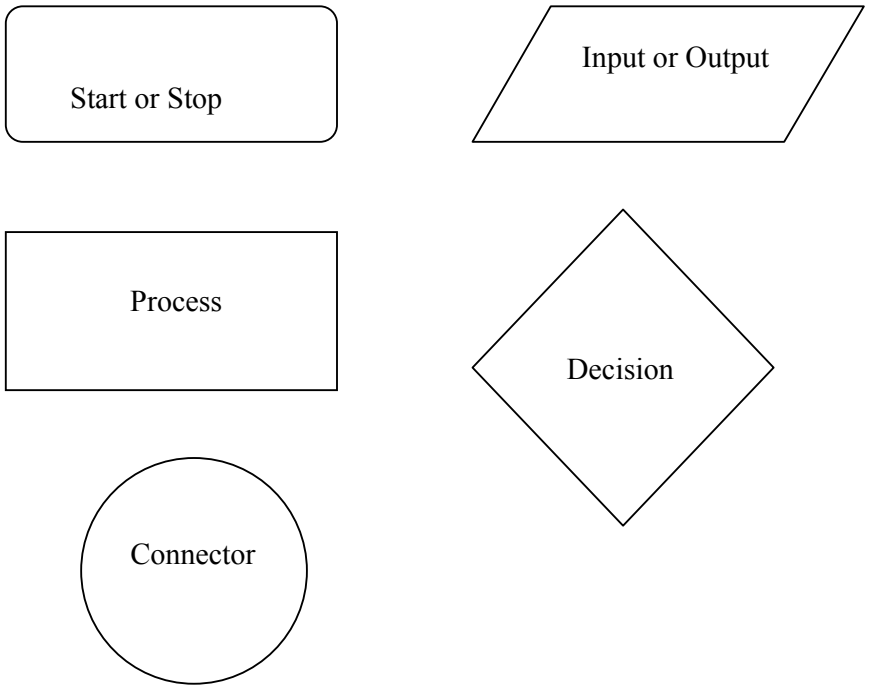
Evaluation:

This module can be solely evaluated using the task sheet supplied will be reinforced when designing programmes to run designed robots. The flow chart configuration will increase pupils computer usage giving them good confidence and experience, (i.e.) selecting and dragging icons similar to that of both programming languages. Further more one language is solely based on flow chart design (RCX code).

## Module 7: Flow charts

Flow charts are used to break down a problem into logical order. The individual steps are placed inside boxes. There are 28 flow chart symbols. Most of these symbols are available in Microsoft word. Clicking on View on the top of your Microsoft word page and selecting Toolbars and then Drawing opens a new tool bar. By selecting AutoShapes will open the flow chart options.

Each symbol stands for a different meaning and when the mouse hovers over these icons there name is shown. Shown below are 5 of the most commonly used symbols.



Stop or start boxes

These symbols are only used at the start and the end of the flow diagram. The flow diagram may diverge and can have two stop or end boxes.

Input and output boxes

These boxes are used to place information or give out responses/answers. They have arrows pointing inwards and outwards.

Process boxes

The function of the process box is to do something. For example information is processed add 3 to the number. Calculations and physical activities should be put it in these boxes.

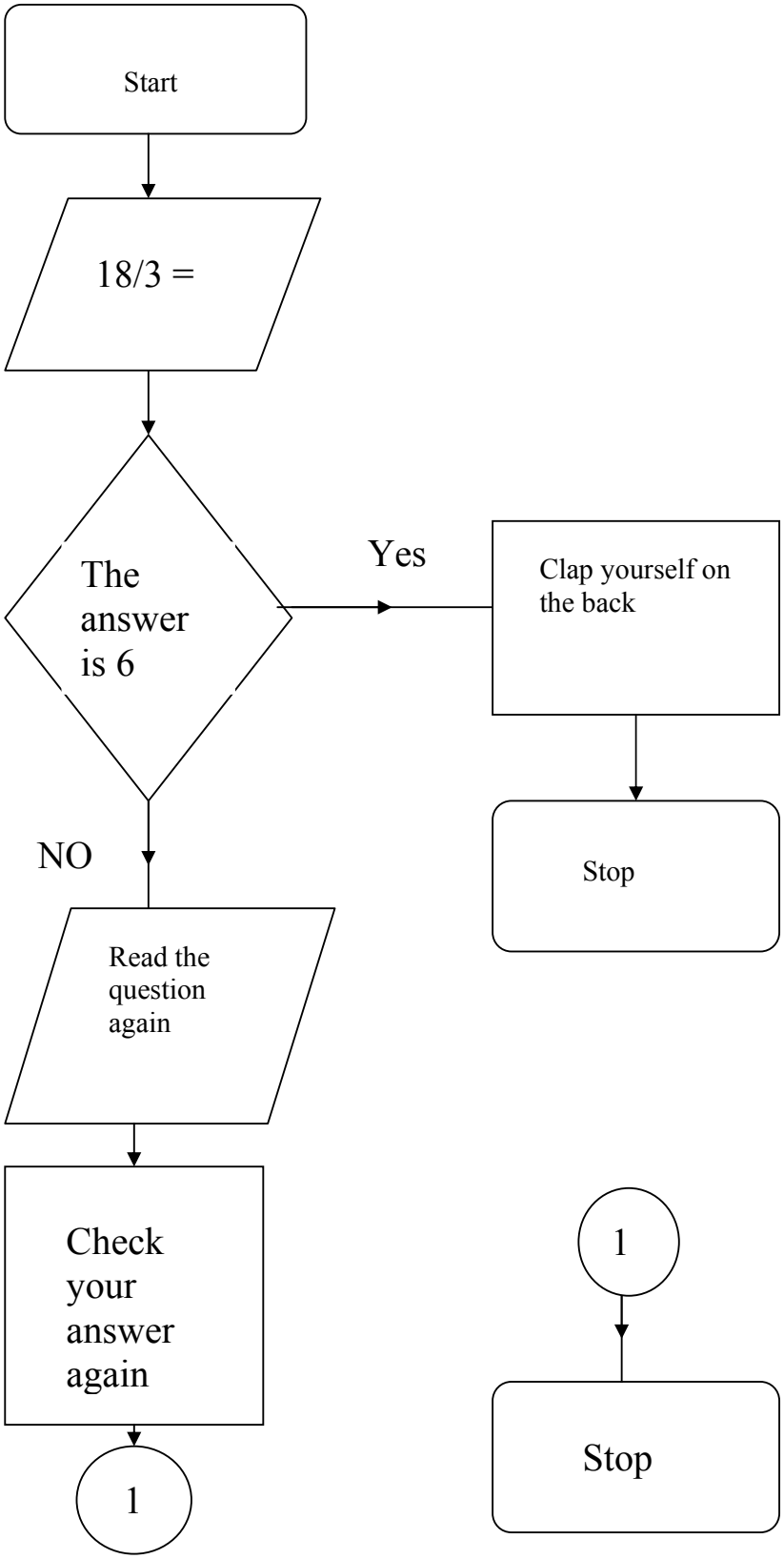
Decision boxes

These boxes can have two responses, either yes or no. There is one input and two outputs in this box. Inside the box is a question and the responses are written on the line exiting the symbol.

Connector boxes

These connectors are used to avoid confusion, for example say the flow chart goes off the page onto another page the end of the last page ends with a connector box with a number in it and then the top of the next page starts with this box with the same number in it.

An example of a flow chart is.



## Computer programming flow charts

Flow charts can be used to help write computer programmes. They give us a logical sequence to follow. There are rules for writing flow charts when programming.

1. Define what you want your programme to do in a logical statement.

2. Consider many options of getting to the solution and pick the most effective solution.

3. Use loops and consider different options such as diverges and merges.

4. Draw a flow chart of the solution. Make sure that the diagram has a flow of logic. Do not be afraid to x out some parts.

5. Draw a final flow chart for the programme. Show it to somebody that has not being involved in the designing process; they should be able to read the flow chart with ease. Clarity is critical here.

## Task sheet flow charts

1. Design a flow chart that will programme a robot to move forward until the touch sensor is activated.

2. Give two reasons why it is a good idea to use flow charts.

3. Design a flow chart that will show a logical method of solving simultaneous equations.

4. A robot is required to follow a black line on a white surface. Draw a flow chart that will keep the robot on the line.

5. What reasons can you give why the decision symbol can only have two outputs, can you relate these out puts to logic (0) and (1).

6. How can you test to make sure that your flow chart actually works?

7. Computer programmes often have flaws in them what are these flaws called, and what's the name is given to the process of solving these errors.

## Task sheet Revision

1. Describe the operating characteristics of a diode?
2. What is the primary function of a flow diagram?
3. In relation to a transistor what do NPN and NPN mean?
4. What do you understand by the term Digital IC?
5. Explain the term feedback in relation to control?
6. Explain what inverting a signal means?
7. What is a short circuit?
8. How do transistors work?
9. Draw a logic circuit that will make a noise only when a weight is placed on a pressure pad and the temperature is below 5 degrees. If the temperature is above 15 degrees and no weight is placed on the pressure pad the noise should also sound?
10. Fill in a truth table for the above circuit?
11. Consider this arrangement, an input of logic 1 is inverted and then is Nanded what is the out put? (logic)
12. Why do we use Karnaugh maps? (explain any construction + example)
13. Robots use human like hands called grippers, how are they powered and what design criteria is used for selecting the power source?
14. A building crane has how many degrees of freedom, what does this term mean and how many are there?
15. What is the difference between weight and mass?

## Module 8: Lego construction (design methods)

Aims:

To introduce Lego fundamentals to pupils with regard to structure, mechanisms and gearing. To increase pupils awareness of construction methods and design capabilities.

Objectives:

At the end of this module pupils will be able to

1. Identify different mechanisms.
2. Use gears and pulleys as a means of rotary motion transfer.
3. Demonstrate building techniques (bracing, 90 degree structures and triangulation).
4. Design and evaluate own and others designs.

Evaluation:

Ability to attempt tasks at the end of module. The standard to what these tasks were completed. The grading criteria for each task should be similar with 70 % going for performance rather than design and aesthetics.

## Module 8: Lego construction (design methods)

Mechanical principals to basic building techniques will be introduced in this unit. These include mechanisms and structures fundamental to building successful robots. These tips and tricks are useful to everybody regardless of what stage you are at in Lego construction, or if you are just starting.

Beams and structures

Beams are the most commonly used building blocks in Lego. They vary in length and can be assembled various ways. The Lego dimensions language is used to tell them apart.

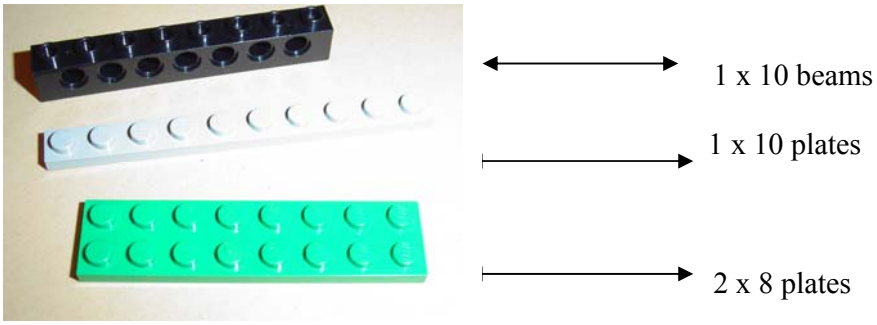
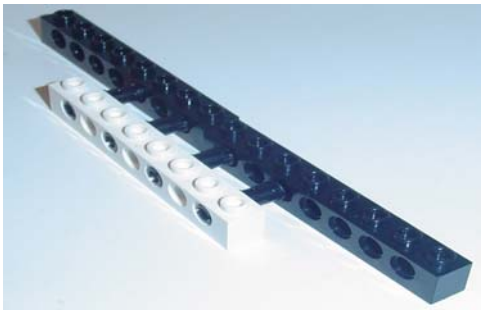



Figure 1

In extending a long frame (black) connector pegs may be used. These should be placed at the ends of the beams to increase stability. This beam can be further reinforced, using plates on top and bottom.



Why do we use the black connector peg and not the grey pegs? The answer to this is that the black peg is larger than the grey peg therefore we use them to lock beams together, and the grey pegs are used when allowing for swing, moving joints.

Plates may be used also; these are just as stable as the previous example and do not use up scarce friction pegs.

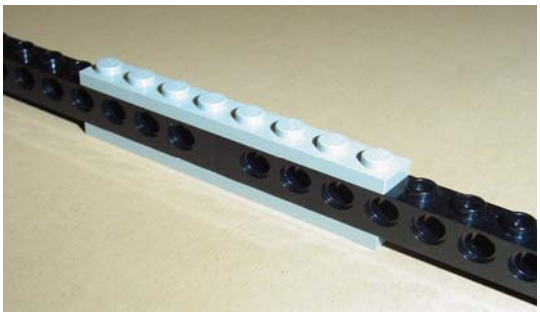


Figure 4

As can be experimented using Lego, triangles are super strong structures. They are used to stiffen up structures. In figure 5 you can see how frames are constructed and how they adjust to loading.

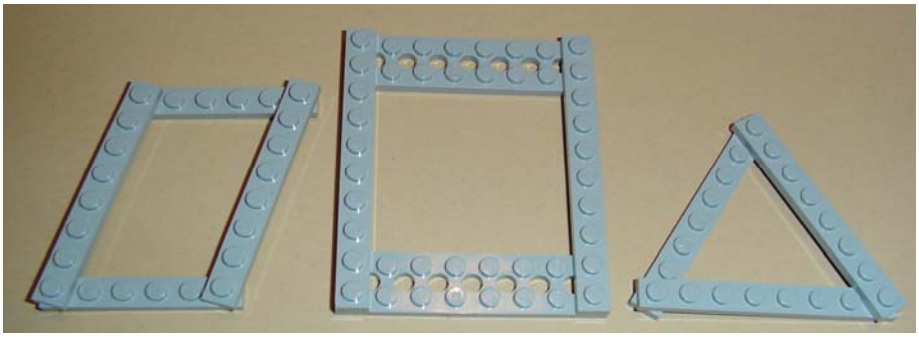


Figure 5

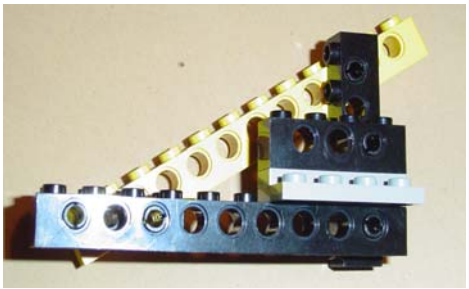


Figure 6

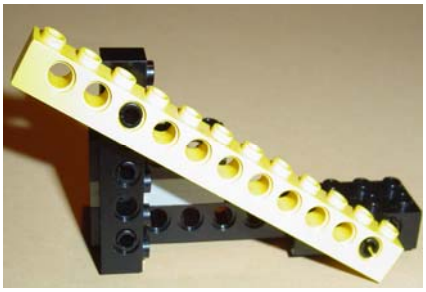


Figure 7

## Bracing

Bracing is the term given to beams that run at 90 degrees to the main structure. These usually vertical beams are connected using the friction pegs. However it may seem easy to do but due to dimensions of Lego some times the holes do not line up.

Lego bricks are 1 unit wide and 1.2 units high. Therefore when you try to connect the bracing to the structure it does not always line up. A plate is 0.4 units tall, so if you use two plates and two beams you get a whole number = 2.0, now you try this out.

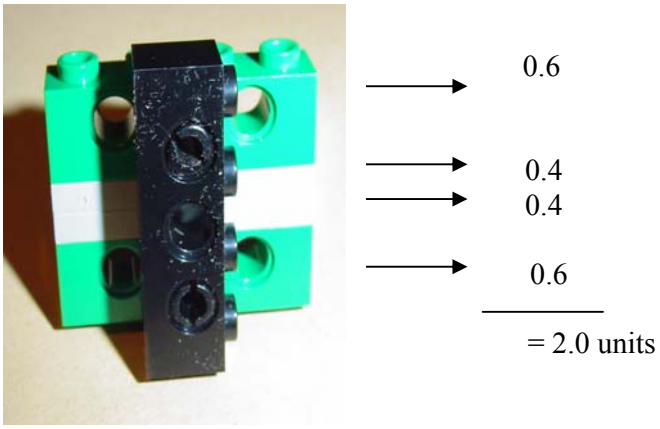



Figure 8

Will four horizontal beams and one plate allow for simple bracing? You do the maths and test it out?

What is the lowest number of beams you can have with out using a plate to allow you to brace them?

Look back at figure 6 and 7 this is bracing involving triangulation.

Try it out,

To build structures at 90 degrees to the main structure many methods are used. The Lego set comes with two 2 x 2 grey angle plates. Also shown are other methods.

Figure 9

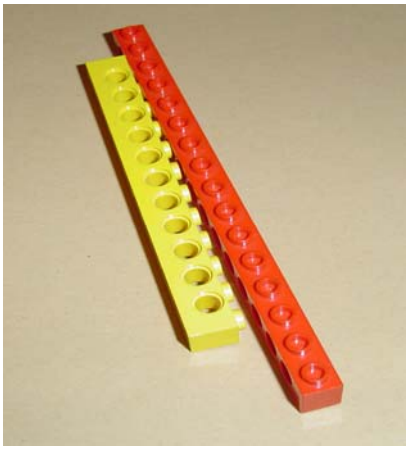


Figure 10

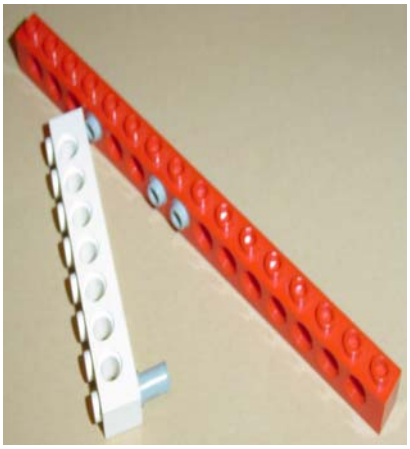


Figure 11

Figure 10 is two beams, one pushed into the other, while figure 11 is using 1.5mm connector pegs and this keeps the friction grips to the out side.

Can you think of reasons why you might need to build at 90 degrees to the main structure?

## Gearing

We have completed the structures aspects of Lego design, now we are going to get our teeth into gears, pullies and various mechanisms.

Gears are needed to convert rotational motion from the motor to a certain place (distance away) at a given speed. The speed of the motor is usually to fast and the torque (rotational power) is very low.

Lego supply various types of gears for different applications. However the standard gears have 40, 24, 16 and 8 teeth, see the connection?

The axles (drive shafts) are various lengths and fit all gears and bushes. We are mostly concerned with gearing down (lowering the revolutions of the driver to the driven). Therefore let's try an example.

If the motor is rotating at 200 revolutions per min and we want to reduce it to 40 revs what gears should we use?

Well 200 is to 40 revs means that we need to reduce it by 160 revs (200-60).

However when dealing with gears we use ratios. What is 200 is to 40.

5:1 now we need a gear combination of ratio 5:1, we can use the 40:8 as this gives us the 5:1 ratio also. But what gear goes on the motor axle, remember you are gearing down.

Yes of course the larger is the driven and the smaller is the driver.

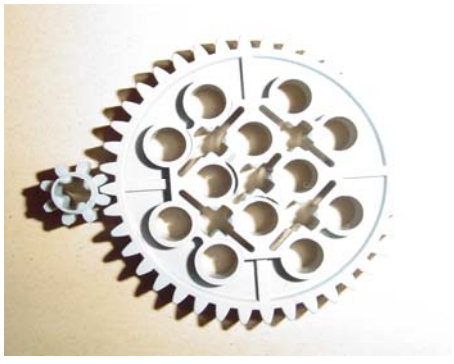


Figure 12

Remember that both bevel and crown gears can transfer rotary motion through 90 degrees. (Fig 13 and 14)

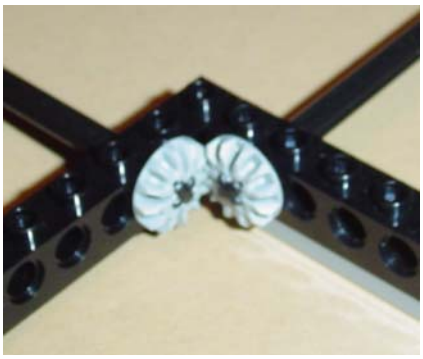


Figure 13

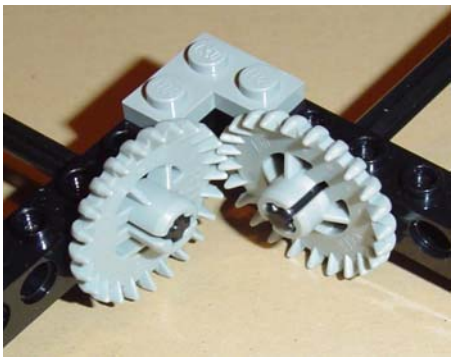


Figure 14

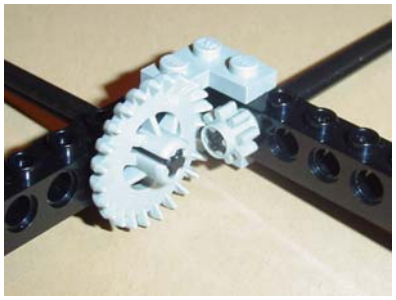
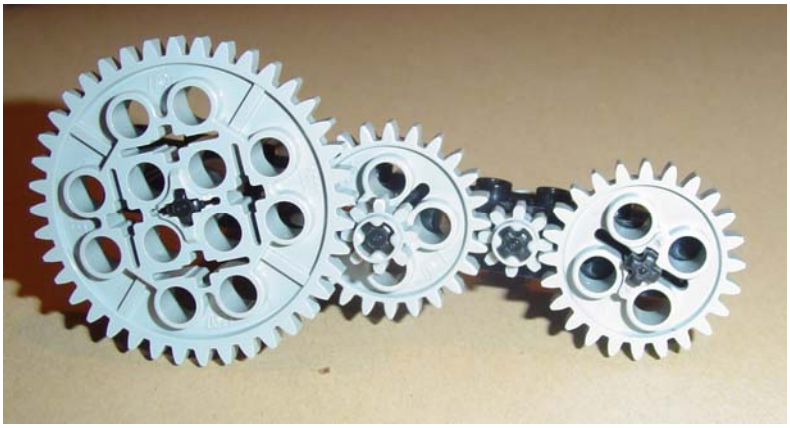


Figure 15

Gear reduction can be used in your design when transferring motion at 90 degrees. Figure 15 shows the reduction 24:8= 4:1 of a speed reduction but does the torque increase.

A compound gear train occurs when a least two gears of different size are on the same axle. These gears are meshing with other gears to the left and right. See figure 16 for a compound gear train. Figure 16



A compound gear train that is most effective in reducing speed while increasing torque is using the largest and smallest gear wheels in the Lego set. Figure 17 shows it construction. The maths is simple, the motor turns the small 8-tooth gear, and it must turn it 5 times to turn the 40-tooth gear once (40/8=5). As there is another 8-tooth gear on the same axel meshing with a 40-tooth gear it needs to turn another 5 times. Therefore 5 x 5 = 25 so the ratio reduction is 25.

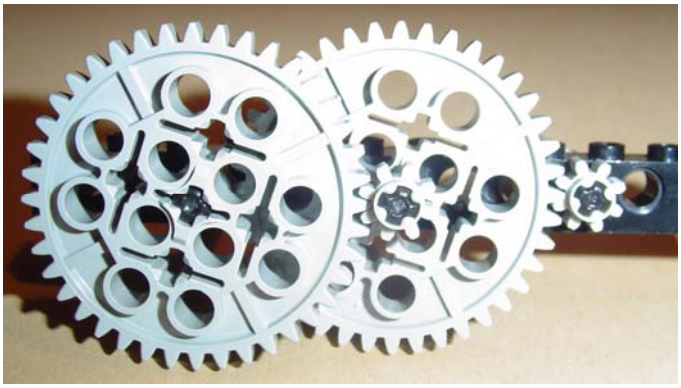


Figure 17

The worm and worm wheel is the most effective method of reducing speed and increasing torque. Rotary power is transferred perpendicular to the main power drive. It is important to note that the worm gear can turn the gear, but to transfer motion from the gear to the worm is unworkable. Try it!

Figure 18

If a spur gear has a radius at infinity (largest distance possible) the result is a rack. The rack travels in a straight line along a plane. It converts rotary motion into linear motion in both directions. It can also transfer linear motion into rotary motion. The power is transferred perpendicular. Figure 19 shows a rack and pinion meshing.

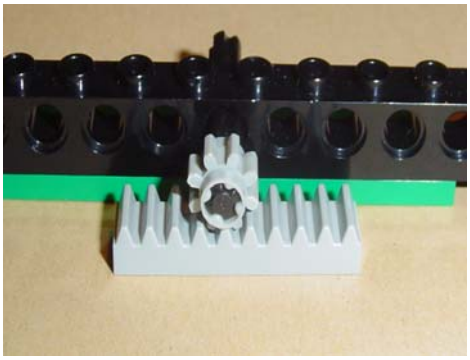


Figure 19

Figure 20

Linear motion can also be transferred parallel to the drive shaft using a rack and worm wheel as is shown above in fig 20.

## Pulleys and belts

The transfer of rotary motion can also be achieved by using pulleys. These allow for flexibility with regard to motor and axle placement, the bands are of various sizes and allow for slip, hence preventing your motors from stalling and wasting the batteries.

Power can be transferred at 90 degrees and output direction can be changed. Figure 21 shows a belt drive transferring motion at 90 degrees and figure 22 shows a simple belt drive. As with gears, gearing up and down is possible with pulleys be cautious not to over expand the pulleys.

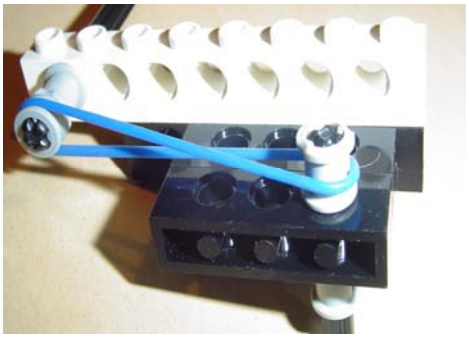


Figure 21

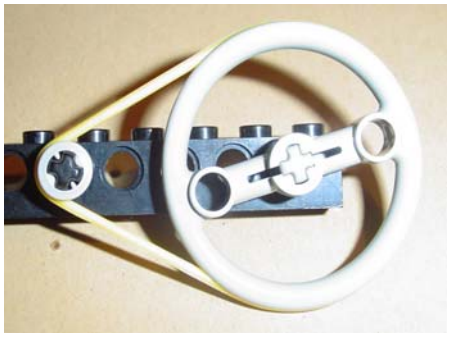


Figure 22

### The differential

This gear assembly is very useful. It has three input or output axles. It is used to allow for cornering as the out side wheel travels more distance than the inside wheel. When assembling you will need to ensure that the centre bevel gear is fitted first and then add on your axles and gears. To study how it works, the best method is to make it and play with it. Figure 23 shows a differential. The differential has many different uses in a robot increase speed, prevents motors from stalling and many more, as you will find out.

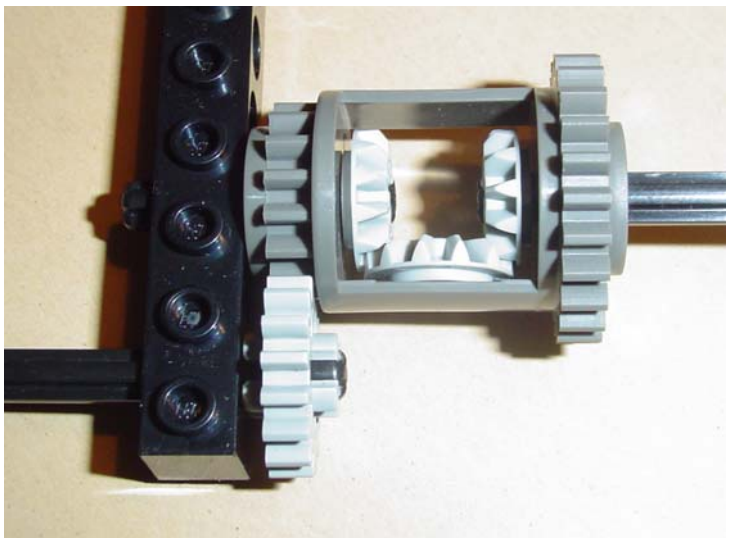
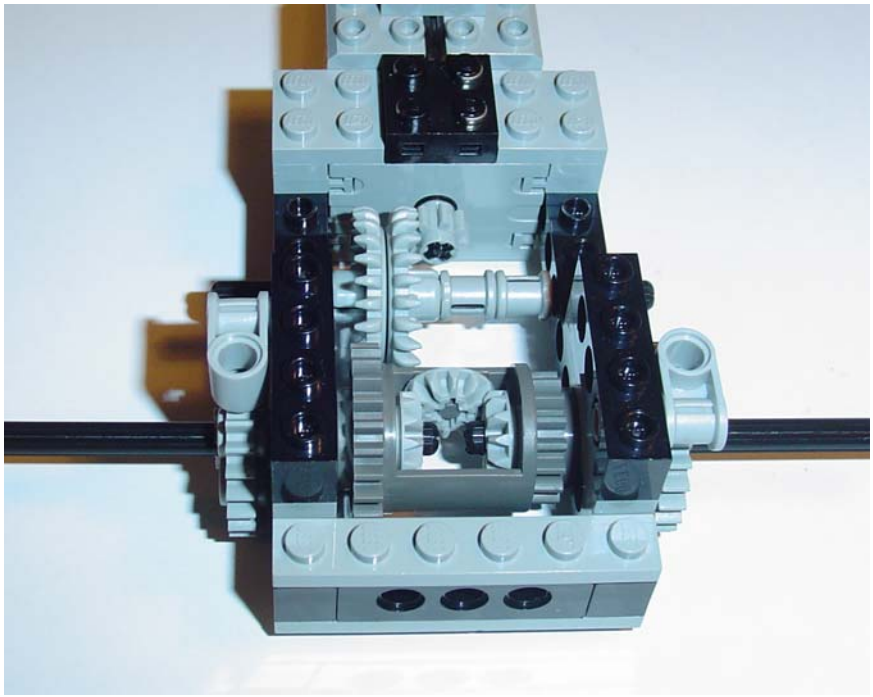


Figure 23

The RIS™ has only two output motors; therefore your robots are limited to there out put actions. However clever use of the reverse and forward direction of motors will allow for greater dexterity, even with this the two output motors can only turn a single shaft in opposite directions. To get one motor to turn two individual shafts is more complex. This is where the differential and two ratchets are used. Ratchets allow for rotation in one direction and prevent rotation in the opposite direction. Depending on the placement of the ratchets and the motor direction only one shaft will turn. If the motor is reversed then one ratchet allows for forward movement while the other prevents movement.

Figure 24 shows a model differential and two ratchets in operation with a motor.



## Lego construction (design methods)

## Task Sheet

1. Using the Lego, make a crane arm 200mm long to carry your RCX through 180 degrees. The crane must be stationary at all times and the mechanisms used should be hand operated.

2. In a group of two, design and make a car that can with stand a fall from two foot. The car must in clued a steering mechanism that can be turned from out side the structure.

3. The design if not already should include a differential.

4. Using a rubber band (not those supplied in the Lego set) as the only means of energy supplied to power a racer that you design, the competition is to see who's racer goes furthers (remember to keep the design simple and light weight).

5. Using the pre-programmed RCX with motor attached design a base that rotates once every minute to hold the winning racer for display purposes. (Do not disassemble the racer) (Gear reduction)

6. Robots travel on legs and wheels design and make a robot that does both when turned over. (The pre-programmed RCX my be used with two motors)

7. Design a linkage system that will lower a book from knee height to the ground and raise a book of lower weight up to knee height.

8. Using the tracks supplied design and make a conveyer belt system that will lift 2 x 1 Lego beams as high as possible and stack them in the horizontal plane, the only holding mechanism is the friction between the belt and the beam.

9. Using the pre-programmed RCX with a motor and touch sensor programmed to turn the motor when the sensor is pressed. Design and make a model of a four stroke engine cylinder. The model should include correct timing of the cam and valves attached. The model does not need to be 3D, for example the piston can be flat plate.

10. With the introduction of the Euro currency, people with poor eye sight still confuse the 5 cents with the 10 cents coins. You are to design make test and modify a mechanism that will differentiate them and stack them at 180 degrees to each other.

11. Design and build a clock that is accurate with a pre-programmed RCX motor at full speed. The RPM of a motor at full speed with good batteries is 200 RPM. Gears will need to be used and the hand should turn 360 degrees every 24 hours.

## Module 9: The RCX

Aim:

To introduce the RCX, motors, touch sensors and the light sensor. To discuss the software and guide pupils through the interactive tutorials provided by the alternate systems.

Objectives:

At the end of this module pupils will be able to:

1. Use basic programming skills.
2. Recognise the different programming languages.
3. Attempt minor programming problems.
4. Time their progression wisely.

Evaluation:

Pupil's as makers of meaning is the philosophy required here. The set tasks should be completed and marked off by themselves. There is no task sheet at the end of this module, this module is the foundation to the programming basics and there is active learning involved, therefore pupils will move on to Pro Challenges quickly.

## Module 9: The RCX

The RCX is often referred to as the brain of the robot. It is the programmable brick. It has three input ports (1, 2 and 3) and three output ports (A, B and C). The L.C.D. screen is the display window that shows relevant information, such as programme number and time. There are four buttons around the screen and they are used for various applications. Figure 2.1 shows an RCX with its 3 sensors and motors.

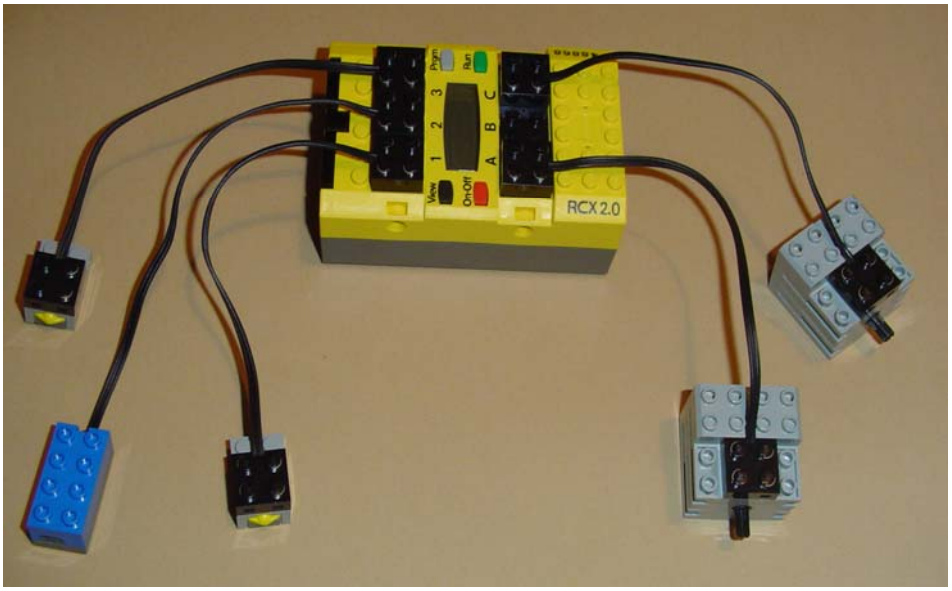



Figure 1

The system is supplied with two motors, two touch sensors and one light sensor. The RCX is programmed using a computer language and then is down loaded using infrared light. This is the interface that links the computer to the RCX. Therefore the tower needs to be insight of the RCX. The RCX firstly needs to down load the firmware, (i.e.) the language of communication. This takes about six minutes and then you are ready to program. Even when the RCX is turned

off it still remembers the programs and firmware, but if the batteries are taken out for longer than 2 minutes you will need to download the firmware again.

### The motors

The motors drain the most power from your batteries. Motors are like generators, place two motors side by side and wire them to each other. Turn the shaft on one of them and look at the shaft on the other side. When programming your motors there is an option to stop them dead called *STOP/BREAK* or cut power from going to them called *COAST/FLOAT*. The first option uses up power as opposed to the last option. The FLOAT method can be demonstrated using the generator model (figure 2.2) and the STOP method can be modelled using a wire loop (fig 2.3)

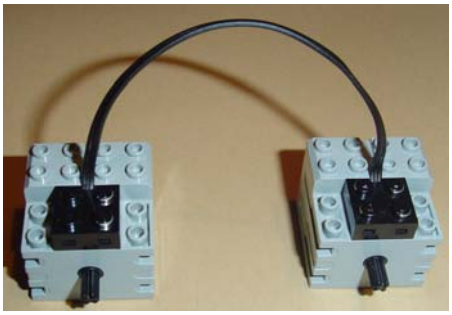


Figure 2

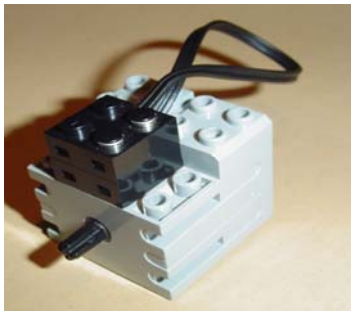


Figure 3

### The sensors

The touch sensors can some times be tricky to work; this is why I advise the use of the 'Try it' option when programming using the Mindstorms software. The wire must touch at least two of the front four nubs with the wire running directly behind. There is also an option of having the touch sensor open, closed or clicked so this provides you with an option when designing. Figure 2.4 shows two touch sensors, which sensor is wired incorrectly?

### The light sensor

The light sensor has its own cable attached and uses power constantly. The front has two L.E.Ds, the clear L.E.D. is a phototransistor that measures the light of the article or desired area, and the second is a red L.E.D. that is used for light reflection. The phototransistor measures light through 360 degrees. To limit the range a simple beam can be placed in front of the sensor, this also prevents the red light interfering with readings. This is demonstrated in figure 5

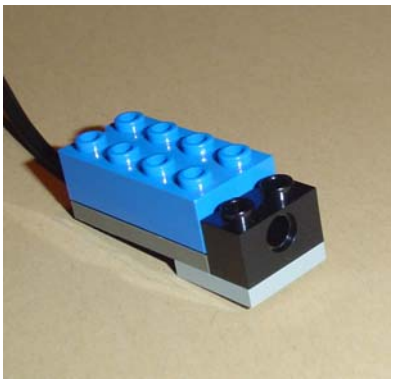


Figure 5

### Installing the software RCX code and ROBOLAB

When installing the software make sure not to plug in the IR tower before it is fully installed. When the software has downloaded you will need to restart your computer. Follow a series of steps to plug in your IR tower and download the firmware. There are two programming languages that we are going to use the second is called ROBOLAB so install this programme after. You will need to download the firmware again. This process should be adhered to as this allows you to programme using whatever programming language you prefer. The only draw back is that the first two programme slots are now locked.

### Interactive missions

The ROBITICS INVENTION SYSTEM™ has built in training missions to the programme. So at this stage you should go through each training mission. The list of missions and sub missions are below, photocopy this page to record what missions you have completed. Speakers or earphones are needed for the audio. Tick the box when complete. Approx time 10-15 min per slot

Name: _____

## Basic

1. This is the RCX- motor, touch and light sensor. ☐
2. Setting up your robot- Building and testing. ☐
3. Programming basics – opening, downloading, editing, saving and accessing help. ☐
4. Using other commands – over view commands, change command settings, coping, scrolling and zooming. ☐
5. Using sensors – touch and light sensors. ☐
6. Controlling program flow – repeats, wait and yes/no, stack controllers. ☐

## Advanced

7. Small blocks – over view and programming with small blocks. ☐
8. My blocks – over view and creating my blocks. ☐
9. Variables – over view and using variables. ☐

## Challenge missions

Now that you have become proficient in the programming and building basics move on to the challenge missions. When you have all the challenges completed called Roverbot, Acrobot and Inventorbot including the 'Take off' missions. Then we will move to a different programming language called ROBOLAB.

Open this up and select training missions. I would advise you to go through all the missions apart from the introductions, as the operation is similar to that of the previous program. Tick the boxes when complete. Approx time is about 8 min per slot.

Programmer inventor Pilot missions

- Programmer pilot 1 ☐
- Programmer pilot 2 ☐
- Programmer pilot 3 ☐
- Programmer pilot 4 ☐
- Investigator pilot 1 ☐
- Investigator pilot 2 ☐

Inventor missions

- Programmer inventor 1 ☐
- Programmer inventor 2 ☐
- Programmer inventor 3 ☐
- Investigator inventor 4 ☐

You will have noticed that the programmer pilot is very easy to use and is similar to the Big Blocks in the RIS™. Therefore we will not be concerned with this programming, however both Inventor and Investigator are very useful and will be used when programming the Pro-Challenges supplied with the RIS™. Therefore in the next chapters we will build each individual robot and program them using both languages.

## Pilot programming (tasks)

The pilot section has limited capabilities but introduces the basics of programming. The controlled environment ensures success and the programme will always complete. Common images/icons are used in this programming language and in both inventor and investigator.

Here are different task for the various levels that the pilot works at.

Task for pilot 1

Reverse motor A for 9.03 seconds.

Task for pilot 2

Place motor A in the forward direction at full power and place motor C at mid-power but in the reverse direction. Stop them after the sensor has being pushed and the input port must be no 2.

Task for pilot 3

Program a car to drive forward until it seeks a dark light at input port 1 and reverse twice as fast as it was travelling forward and stop once the rear bumper is touched at input port three. Make the programme repeat itself continuously.

Task for pilot 4

Program the RCX with 2 motors attached, 2 touch sensors and the light sensor.

(Step 1) Start with motors A and B at full speed, and then programme that when the touch sensor 1 is touched.

(Step 2) The motors run at power level 3 in the reverse direction for five seconds.

(Step 3) After the 5 seconds have elapsed the motors should travel forward at the lowest speed possible until the light source is greater than 60%.

Task for pilot 4

Design a toll bridge for a Lego car; the brief requires the RCX to be told when the car is present and when the light sensor is above 60%. The barrier rises until the car has passed and then lowers after the car has passed a certain point on the road. Keep your design effective and add as many steps as possible.

Task for pilot 4

Design a traffic indication system. The system must tell the car drivers the driving conditions ahead. At least three driving conditions most be shown. The signs need to be lowered over the cars as they drive.

Task for pilot 4

Design an environmentally friendly window guard. When the light source is less than 45% the guard/shutter will lower to keep the heat inside. When the light is greater than 55% the guard/shutter moves back to let the light in.

Task for pilot 4

Find out how to play Beethoven and Hall using the pilot level 4.

## Module 11: Introduction to programming and problem solving

Aims:

To introduce pupils to the programming languages. Using a helicopter as a means of motivation and problem solving hence becoming proficient in simple programming.

Objectives:

At the end of this module pupils will be able to:

1. Use basic inventor programming skills.
2. Become familiar with *Task Splits*, *Loops* and *Variables*.
3. Attempt minor programming problems.
4. Solve design related problems.

Evaluation:

The ability to programme the helicopter to perform different tasks. There problem solving methods (logic statements or flow diagrams). Willingness' to test modify and change incorrect programmes. Helicopter design features and structures.

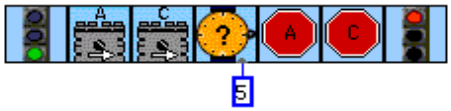
## Module 11: Introduction to inventor programming and problem solving

The design brief is to design a helicopter that has two independently powered rotor blades. The design should include two touch sensors and room for the RCX to be held with in the cockpit. This design is your own and should fulfil the brief.

We are now going to programme the helicopter to do different things involving the rotors.

**(1)** The first programme design brief is to turn on the motors for 5 seconds in the forward position. Both languages are demonstrated here. A flow chart or logic statement can be stated or drawn here.

"Turn on motors A and B, wait 5 seconds, turn of motors A and B"

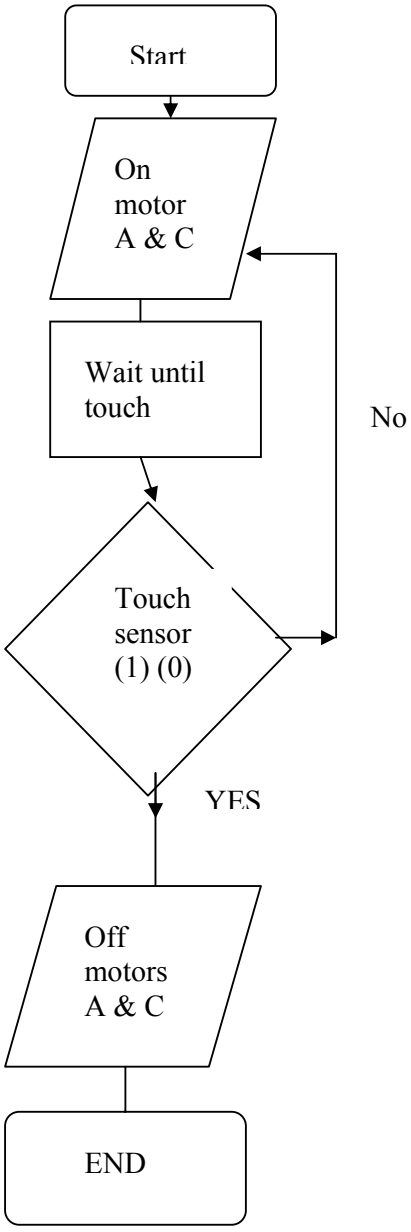




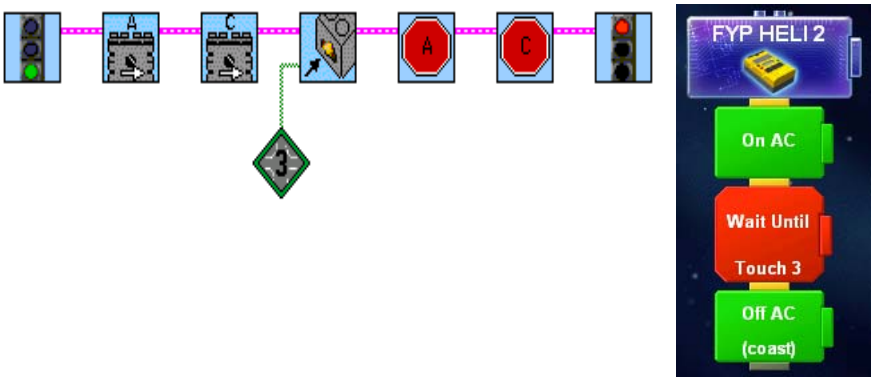
Now that you have programmed your helicopter rotors to rotate for 5 seconds, programme your helicopter to rotate its rotors in the opposite direction for 8 seconds.

**(2)** Now that you have programmed the two motors to turn off using time. Programme your helicopter to turn off using the touch sensor at input port 3.

A logic statement or flow diagram can be used to solve to problem.

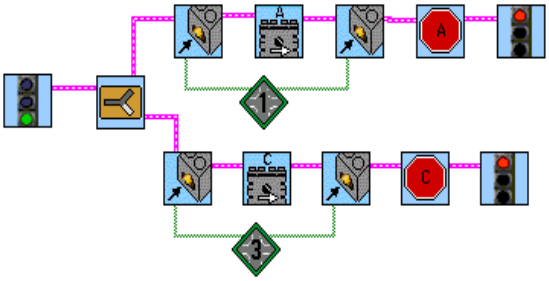



Now programme your helicopter to complete the task.




Once you have programmed your helicopter to successfully stop when the touch sensor 3 has being touched, modify your programme to start when touch sensor 1 is pressed and continue on the same as the problem above.

**(3)** This design brief requires you to power each motor independently of each other. Touch sensor 1 should be able to turn motor A on and off, while touch sensor 3 should be able to turn motor B on and off. A logic statement should be avoided here so use a flow chart. The flow chart will divide and may or may not join again before the finish.



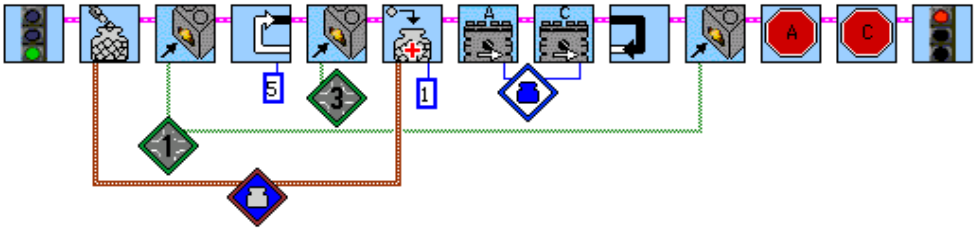


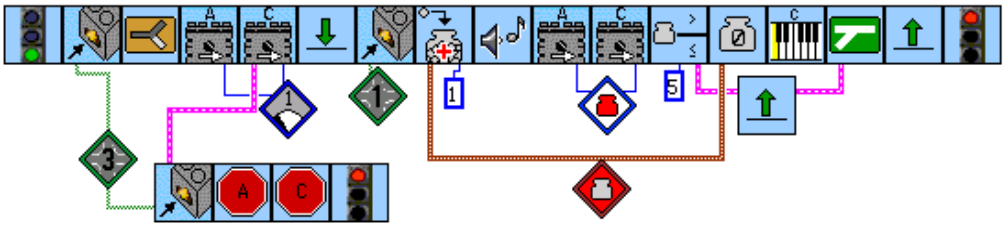
If programming using the RCX code why is the *Wait For 1 second* used?

Make the programme repeat indefinitely using either programme. Is it possible to *JUMP* and *LAND* over a *Task Split?*

**(4)** The brief is to programme your helicopter to have to ability to accelerate. The touch sensor input port 1 should be used to start and stop motors A and C while the other touch sensor is used to increase the speed of the motor.

(Only capable using ROBOLAB)





What is the difference between these two programmes and which programme solves the brief best?

You should now design a programme that will start the motors in full speed and de-accelerate them to stop. Each time they de-accelerate a mark (5-0) a sound should be played. The programme must be able to repeat itself at least 4 times.

**(5)** Programme your helicopter to use the two sensors one for acceleration and the other sensor for de-acceleration. It is up to you how to start and stop your programme.

## Task sheet RCX Programming

1. Design a program that will run motors A and C, reversing motor A after 8 seconds. Then, it will stop motors A and C when touch sensor 2 is released.

2. Design a program to run motor B when light sensor 1 is less than 35%. When light sensor 1 is greater than 35%, it will reverse motor B for 8 seconds and then stop the motor.

3. This program will run motor C while touch sensor 2 is released. When touch sensor 2 is pressed, motor A will reverse for 2 seconds and stop.

4. Create a program to run motors A and C until touch sensor 2 is touched, and then will reverse the motor for 3 seconds and then all motors stop.

5. Modify the above program to run motor B after 4 seconds when touch senor 2 is released and then reverse motor B for 5 seconds when touch sensor 2 is pressed. This program should continually check the touch sensor's state by jumping to the beginning.

6. Program the RCX to run motor A for 3 seconds, and then wait for touch sensor 2 to be pushed. Then, it will reverse motor A. Then it will wait for touch sensor 2 to be released, and continually run this set of operations by jumping to the beginning of the program.

7. Program the RCX in inventor 3 to turn on motor A after 3 seconds when light sensor 3 is greater than 42% and then reverse motor A for 3 seconds when light sensor 3 is less than 40%. This program will need to check the sensor's state after completing the course of program.

8. This program will wait for 4 seconds and run motor B. Then, it will wait for a decrease of 20 in light sensor 3, and then reverse motor B and run motor A for 1 second.

9. Program motors A and B to turn on at the start of the program and wait for light sensor 3 to be less than 27%. The program will reverse motor A for 8 seconds and then all motors stop.

10. Program the RCX to turn on motor A when the light sensor is less than 40%. When the light sensor reads a light value above 50% the motor B should start and A and B motors should not be on together.

11. Design and test a program that will use one touch sensor and turn on motor A when the touch sensor is pressed, when the touch sensor is released motor B should be turned on. A and B motors should not be on together.

## Module 12: Smart programming

Aim:

To introduce pupils to the concepts of smart programming and extra the capabilities in both programming languages.

Objectives:

At the end of this module pupils will be able to:

1. Programme their robots smartly.
2. Use the copy and paste functions.
3. Create and use *My Blocks* in RCX.
4. Assemble and test *Subroutines*, *Event Modifiers, Sub VIs and Music.*

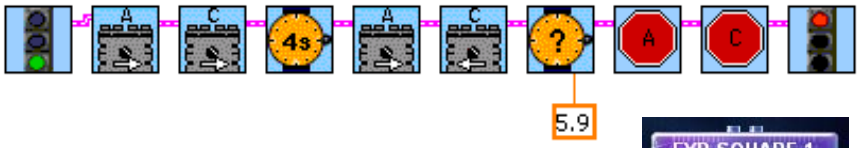
Evaluation:

The evaluation of this module can be both practical and theory based. Pupils can be given a specific task that involves the use of the new learning. This smart programming module will stand to pupils when programming and problem solving when dealing with certain tasks. Programming on the computer with the testing and modifying of a particular robot should be encouraged.

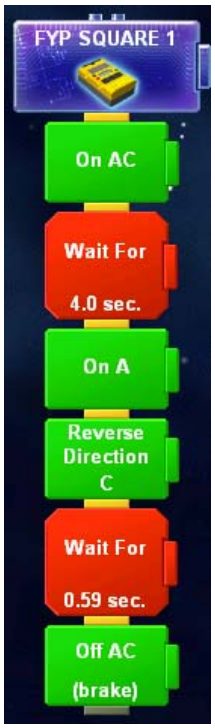
## Module 12: Smart programming

The design brief is to construct a land veicle that has the ability to turn forming a square. The robot can use any method it sees fit for locomotion. It is important to note that the programming times may differ and should be calculated before final programming.

**(1)** Design a programme that will complete a quarter square (i.e.) travel forward a certain amount and then turn 90 degrees.




**(2)** Now programme your robot to travel in a square. It must finish in the same place as it has started.

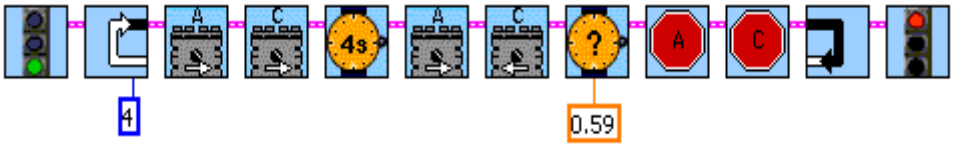


Did you programme your robot this way?


If you used this programming method did you use a copy function or did you select all the icons independently. This type of programming that repeats itself over and over again should be avoided. The use of loops or repeat can reduce programming time and prevent the programme screen becoming too full and long. The fewer icons on the screen can make reading the programme a lot easier.


Programme repeats itself 4 times.

A simple *Loop* function in ROBOLAB with the amount of times (4) wired to the start of the loop can prevent a lot of programming.

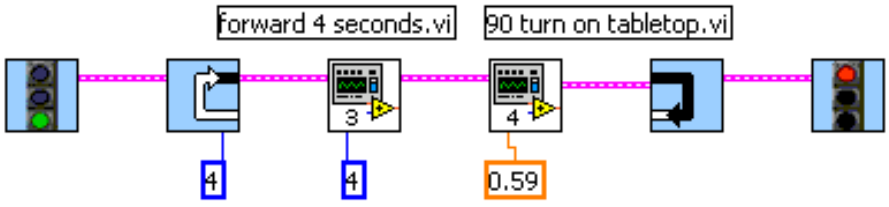



Using the RCX code a *Repeat block* can be used and this block is modified to allow the programme repeat its block contents for the required amount of time, 4 in our case. If this block is used it can be closed to allow for easier programming and other commands can be preformed after the repeat function has carried out its repeating. These extra commands can be attached on to the same *Stack Controller* or the programme block.

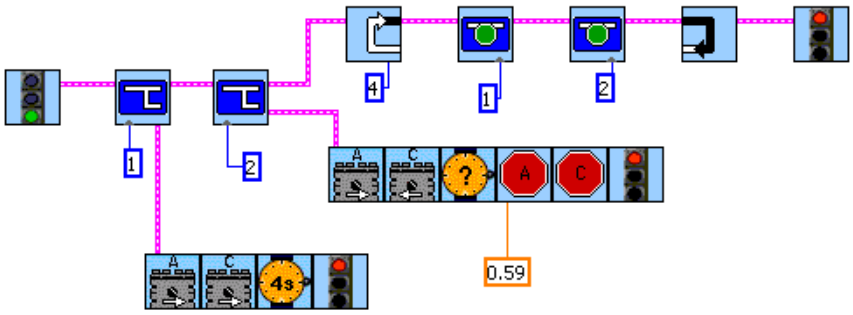
This programming can again be minimised with the use of the My Blocks in RCX and SubVI in ROBOLAB.

To create a *Personal My Block* just select *Create New My Blocks* and name it something that will remind you of the programme inside. I have selected to use two My Blocks one for the forward and the next for the turn as they then can be used in other programmes.


To create a Sub VI in ROBOLAB select the programme that you want to include in your Sub VI and click on Edit and at the bottom of this menu is Create Sub VI. I have selected two different programme areas again. The same as the RCX selected areas. To make programme reading easier these Sub VIs can display what you have saved them as. This is done be selecting (right clicking on the icon) and clicking Visible items and then labels. The importance of the saving name is shown here. The save name should reflect the programme.

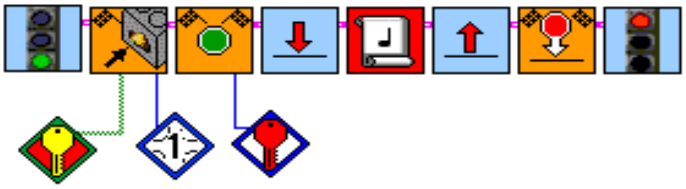


ROBOLAB™ offers another way of programming using *Subroutines.* The *Subroutine* firstly has to be created and stopped using a red traffic light. From the diagram below you can see that there are two *Subroutines* (1) and (2). Then the programme enters a *Loop*, which will run both *Subroutines* four times. This type of programming can become useful when a *Subroutine* needs to be run three or four times in a programme. The *Create Subroutine* can be referred to many times and is a good method of displaying programme parts and it avoids excessive programming.



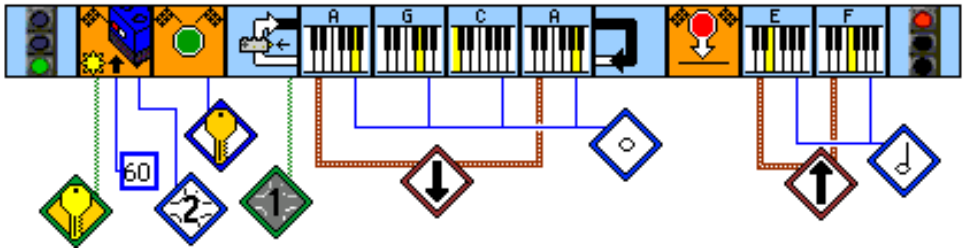
ROBOLAB also offers another method of programming that is not available with RCX Code called *Event Modifiers*. These icons can prove invaluable when programming. An *Event Modifier* can modify the written programme when a preset event occurs. For example the programme can be set up to modify when the touch sensor is touched.  The event must be set before you start to monitor for this event. A start monitoring icon is shown here.  The start-monitoring placement in the programme can be important and in some cases you can get a programme to move out of a never-ending loop.

The event-landing icon is  and is usually placed somewhere further down the programme. To modify an event keys have to be used from the modifiers sub palette.  The diagrams show the value of blue event and the blue event modifiers. Here is an example of how an *Event Modifier* is used.



This programme set up a red event that if the touch sensor is pressed at input port 1. The event monitoring has started by the second icon and then a music loop starts. If the touch sensor is untouched the music will continue. But if the sensor is pressed the event lands on the second last icon and then the programme stops. This is a rather simple method of using an *Event Modifiers*.

The programme shown below has a yellow event set up that involves a light sensor at input port 2 at a threshold level of 60%. The third icon from the right hand side starts to monitor for this event. Then the programme only loops if a touch sensor at input port 1 is pressed constantly. If the touch sensor is pressed will the loop occur but if the light sensor reads a value greater than 60% the programme will move out of the loop and play keys E and F at half notes up an octave regardless of the touch sensor. This programme displays how task priorities can be produced; the event modifier can over rule the state of the touch sensor.



## Module 13-24: Robot design briefs

Aim:

To further develop pupil's problem solving skills by posing robot design briefs.

Objectives:

At the end of these individual design briefs pupils will be able to:

1. Problem solving using various techniques to assist in answering the problem.
2. Design, test and modify both Lego construction and the robots computer language.
3. Analyse different constructions for the same task and select the most appropriate construction for that task.
4. Further develop mental appreciation for technology by studying robotics.
5. Contribute to pupils work skills such as group/project work, independent study and research.

Evaluation:

The evaluation of the following design briefs can be broken down depending on the specific area of focus. Evaluation of each brief should be explained to each pupil for example 33% might go towards design of the robot, 33% towards programming and a 33% to project write up.

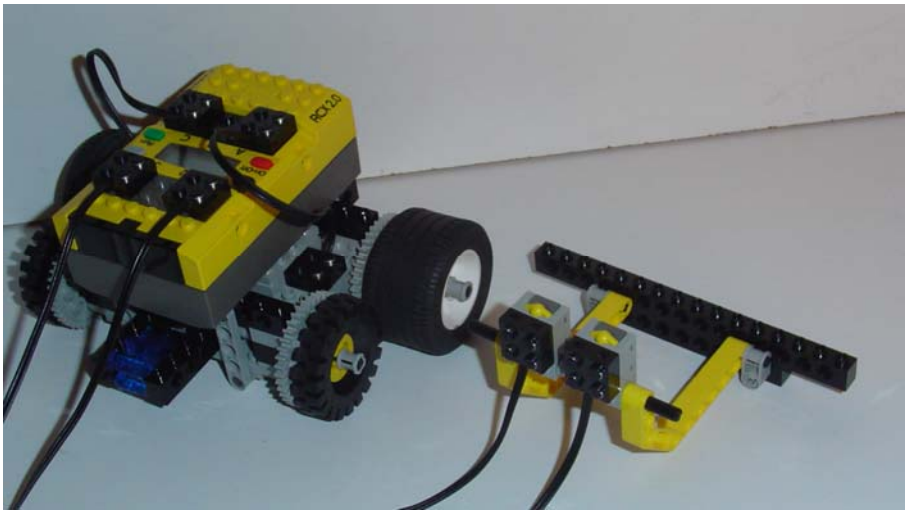
## Module 13: Remote control

The design brief:

1. The robot has to go forward by you using the touch sensors.
2. It must be capable of turning.
3. It may have legs or wheels for locomotion.
4. It should be able to continue on forever providing it meets nothing.

The brief out lines the main functions of the robot, now we need to break these down so we can program it successfully.

It is a good idea to list the items that may be needed to meet the task, for example two motors and two sensors are required, and the light sensor is not. It will need either wheels or legs, so I am going to keep it simple and use wheels. We can use the plans and ideas from the Constructopedia provided with the set to build the robot. Figure 1 shows my robot with the touch sensors attached to the long connections to allow for comfort when operating.



Now that you have built the robot it is time to program. I am going to program the example using the RCX code and then using ROBOLAB™ code.

As stated before it is important to have a clear understanding of what you want your robot to do and under what conditions/circumstances. I find a written plan like the one shown to be most helpful. It makes the flow diagram clearer and it irons out any problems that you may be having.

| | |
|---|---|
| *2 motors* | |
| *2 touch sensors* | |
| *Sensor @ input port 1 for motor @ A* | |
| *Sensor @ input port 3 for motor @ C* | |
| | |
| *Sensor 1* | *Sensor 2* |
| *When pressed* | *When pressed* |
| *On motor A* | *On motor C* |
| *Until released* | *Until released* |
| *Off motor A* | *Off motor C* |

The flow diagram can be written once the plan is completed. The programming language first used is RCX code a sample program is shown. The areas of interest will be highlighted and made clearer as we go through the program. Due to the spread of the programming system I will take a left to right sequence the program is also on the CD supplied with this package, therefore why not open it and follow the screen also.


With all programs the name they were saved as will appear here.

Remote control

When sensors are selected they from across from the title block (RCX). They have to be satisfied first before anything happens.

Figure 2


When the touch sensor 1 is pressed
The following must happen as you have satisfied the requirement.

Motor A turns on
You will want the motor to keep running when you have pressed the sensor

The wait until function has to be changed to release before you add it to the list of blocks.
When the sensor is released the A motor is turned off.

Figure 3


When the touch sensor at input port 3 is pressed the following will happen.

The Motor connected to output port C will turn on.

The RCX keeps the motor on until (touch) the sensor is released.

This then turns of the C motor.

Figure 4

This is the first program that we used that needs to change the event of the red (*Wait Until* touch 3) block before we add it to the program. If you try to add a *Wait Until* block to the program with out selecting to change the event the systems tells you "This block cannot check for the same event as the sensor watcher in which it is contained." Even though the block reads Touch when you have complete changing the event to Release, as shown in figure 5.

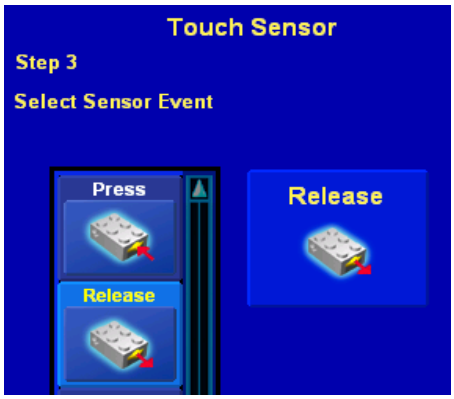



Figure 5

Now we are going to program the same design brief using ROBOLAB™. We are now going to progress to Inventor 3 to program the robot to achieve the brief.

Inventor level 3 allows for more *Structures* than Inventor 2 and there is a music function also that allows the RCX to play music notes.

Now it is up to you to program the robot to perform the same task as the previous program. Try it out yourself, and if the robot meets the brief well done and move on to the next module. If you were having problems in programming this robot it would be a good time to revise the previous Inventor challenges set. On the next page is a sample program for the robot the programme is supplied on the CD under programmes.

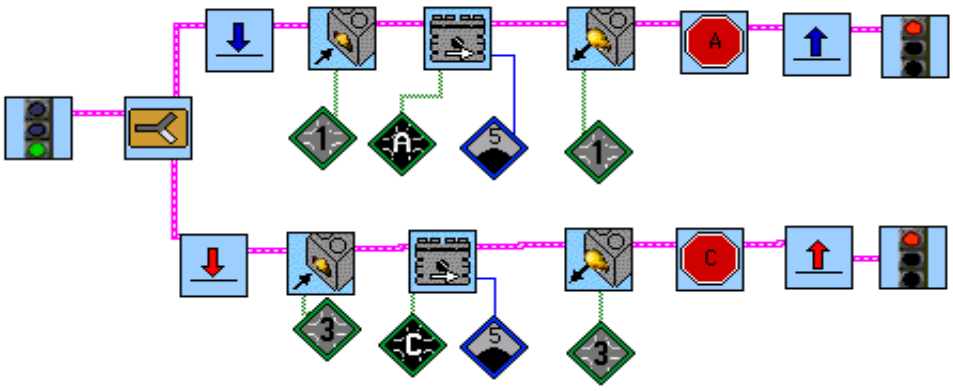



Figure 6

Shown above is a sample program for the remote control robot. This robot needs to be programmed in the Inventor 3 programme as you can see there is a task split straight after the green traffic light that allows the program to branch off and run multiple tasks simultaneously. Also needed are two jump and land structures that allow the programme to repeat forever. The program after this is self-explanatory.

## Module 14: The Line Follower

Design and program a robot to follow a black line against a white surface. The line should be 10 –15 mm thick. Only one light sensor is used in my programmes however if you have two sensors why not use the two of them. The robot can be of any design you prefer however wheels are most reliable and they keep the sensor at equal height from the surface leading to a better reading of the lines. The sheet provided may be used as the line to follow.

The most important aspect of this program is to tell the robot what to do when it goes of the line and what to do when it is on the line. You should check to see the light sensor readings before you program your robot. For my program the black line had a reading of 42% and the white line a reading of 57%. This may vary considerably depending on the line and on lighting in the room. When using only one light sensor the robot will need to hover the black line then turn away and hover the white area and then turn a way from this to go back to the black line.


Figure 1

The program that I designed is quite different to the program supplied with the RIS™ therefore I will discuss my programme with you and the program designed using the ROBOLAB™ code. You can access these codes on the CD. Check to see do the two programmes do the same things and are they meeting the brief.

**Line follower using RIS™ code**


The program name is called whatever you save it as.
LINE FOLLOWER
There are no blocks added to this, as the program will be watching the sensor and taking action based on its conditions.

Figure 2


If the light sensor at input port 2 sees a light between 55-100%.

The motor at output port C will turn on.

The motor at output port a will coast off.

Figure 3


If the light sensor detects light at 0-45% dark light (black line), at input port 2.

The motor connected to output port A turns on.

The motor connected to output port C coasts off.

Figure 4

As can be seen form the above program only one sensor is used however if you have two sensors integrate both of them into the program and see will the robot perform the task any better or quicker.

When looking at the light sensor readings, there is a difference of 10%. 0-45% and 55-100% the reason for the difference is because it lets the robot travel that much quicker as it creates a threshold that needs to be achieved before any action takes place. For example if the sensor is reading 48% what happens, the motor that was active now starts to coast for a while as the light source is neither dark nor bright enough. By the time the motor is finished coasting the sensor will have reached on of the variables and will go through the program again.

**Programming the line follower using ROBOLAB™**

The ROBOLAB™ Inventor 3 programming code was used to program the line follower. Any Inventor level lower has insufficient functions that will allow it to be programmed. The program it self uses a specific task split, called a *Light Sensor Fork*. This allows the program to take either of two actions based on the light sensor reading. The program that I designed is below and is explained after figure 6.

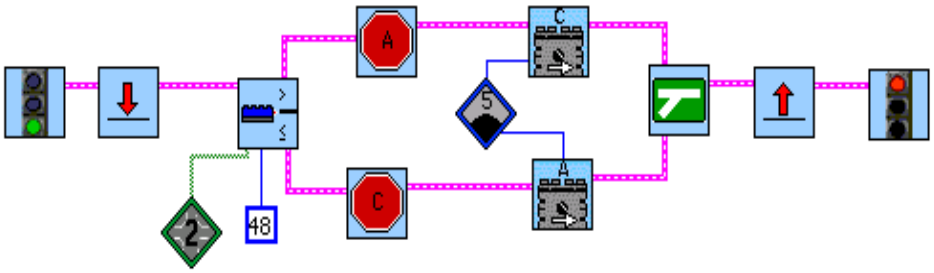



Figure 6

The *Light Sensor Fork* threshold is set to 48% at input port 2. If the light source is greater than 48 then the motor connected to output port A is turned off and motor C is turned on. Then the program meets a *Merge Fork* and jumps at the red jump to land back at the start.

Then if the light sensor gets a value less then or equal to 48% the lower route is taken. The C motor is turned off and motor A is turned on to full speed (the 5). The program is never ending and keeps looping until the RCX settings (time down) is reached and then turns off.

## Module 15: The Art Robot

The design brief for the Art Robot is to design and program a robot to draw lines and curves using attached pencils. A light sensor is needed to follow lines that have already being drawn. It is important that the driving base should allow the Art Robot to run smoothly. The robot in figure 8.1 has the capacity to hold up to three pens. Patterns can be made using different programmes and the repeat function. It would be a good idea to read through the task sheet at the end of this module now to see what's expected from you. The design for the robot in figure 1 can be found in the RIS™ interactive pro-challenges. However you are encouraged to design your own Art Robot.


Figure 1

Programming this robot is totally at your own discretion. You may want to try out different motor speeds and use the repeat functions. Programming the Art Robot using either code/language is good enough and my examples will be talked about briefly in both languages.


The program for the Art robot that I designed is quite simple.

The power is set to full for the motor connected to output port A and it is then turned on for two seconds in the forward direction.

The direction is reversed and this is going for one second.

Then motor A is turned of by coast.

Motor power to output C is set at 7.
The motor is turned on for a total of three seconds and then coasts off.

This whole routine is repeated for 8 times and then the program ends.

Figure 2

**Programming using ROBOLAB™**

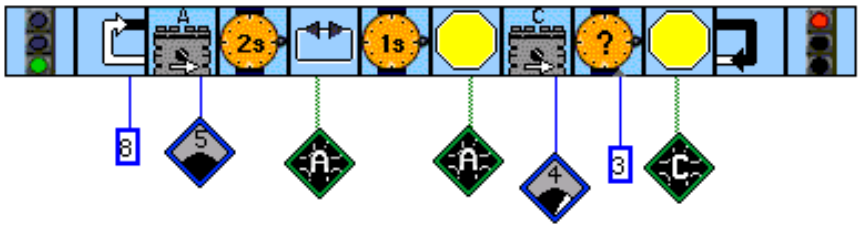



Figure 3

Programming the Art Robot using ROBOLAB™ is quite simple also but does not carry out the exact same motions as the RCX code due to the power variables in the both languages are different, there are five variables in the ROBOLAB™ language while there are five in the RCX code. This is not a problem as the desired effect is achieved. You should move on to the task sheet now and attempt the tasks.

## Module 15: The Art Robot

Task sheet

1. Using one pen in its holder, draw the smallest circle possible.

2. With the same circle bisect the circumference into 6 equal lengths.

3. Programme the robot to draw a straight line of 60 mm long.

4. Using the light sensor attached to the robot, programme it to enlarge a previously drawn square by the distance it is from sensor to the pen tip.

5. Using dead reckoning programme the Art robot to draw a square with the smallest possible corner radius of height 200mm and length 300mm. Surface teat results will need to be taken for this task to be successful.

6. What is the radius of a circle, that has both motors going in the same direction but one motor has twice as much power as the other. (To do this task correctly power levels 8 and 4, 6 and 3, 4 and 2, 2 and 1 should be used with the data saved in an Excel document.)

7. Design a logo for an American corporation; they are attempting to re-ignite the steel industry in Ireland with recycling as a major theme. You have to design a programme/s to complete your logo. (creativity and dead reckoning)

## Module 16: The fridge guy

The design brief is to create a robot that will react to different forms of lighting. For example the robot could be placed in a dark room (fridge) and can record the amount of times the door has being opened. The robot wears Lego sunglasses and these are lowered when the bright light is shining and raised when the light is dark.

Make the robot play notes if his hand is pushed. If his hand is pushed, only the glasses should lower when the light is on and the counter should not register this and not add one to the light count.

My solution to the design brief is as follows. Figure 1 shows the physical model as shown in RIS™ interactive software. These plans can be used however it is better to design your own models.

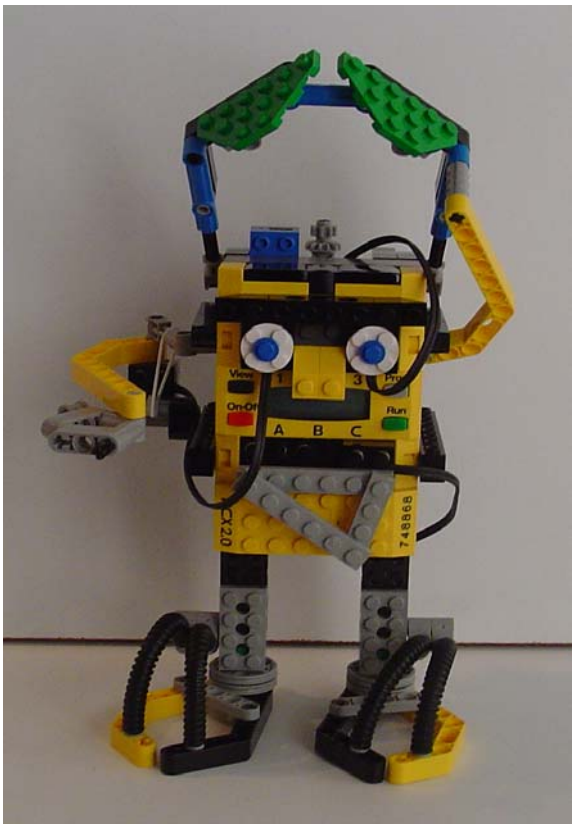



Figure 1

**RCX language**


Figure 2

As can be seen form the above program that the RCX LCD displays the value of the light count. This light count is a new variable created for the program. It is good programming practice to set counters and variables to 0.0 at the start. One light sensor is used to operate the glasses. The sensor is watching for different percentages of light. You can see that big blocks are provided in the RIS™ (lower glasses and raise glasses). This helps when programming as you do not need to worry about what motor and what direction. 1 is added to the counter each time the light is greater than 64% therefore when the door opens the light turns on and the glasses come down and one is added to the counter. When the door closes the light turns off and the glasses move up. This solves the first part of the brief however we need to allow for the handclap. The handclap is your method of telling the robot not to count. It responds to your clap by beeps/music and it leaves the light counter untouched.


Figure 3

The program starts of the same as the original program but alterations have to be made for the touch sensor on the arm of the robot. The sensor waits until the light percentage is greater than 60% and then gives you 3 seconds to hit the hand. If the touch sensor is released (handclap) then the program takes the Yes option. It beeps and lowers the glasses then waits until the light percentage decreases to below 40% and then the glasses rise.

However if the touch sensor is not released the No route is taken, the glasses lower and wait until the light 40% is reached and then the glasses rise.

**Programming the fridge guy using ROBOLAB™**

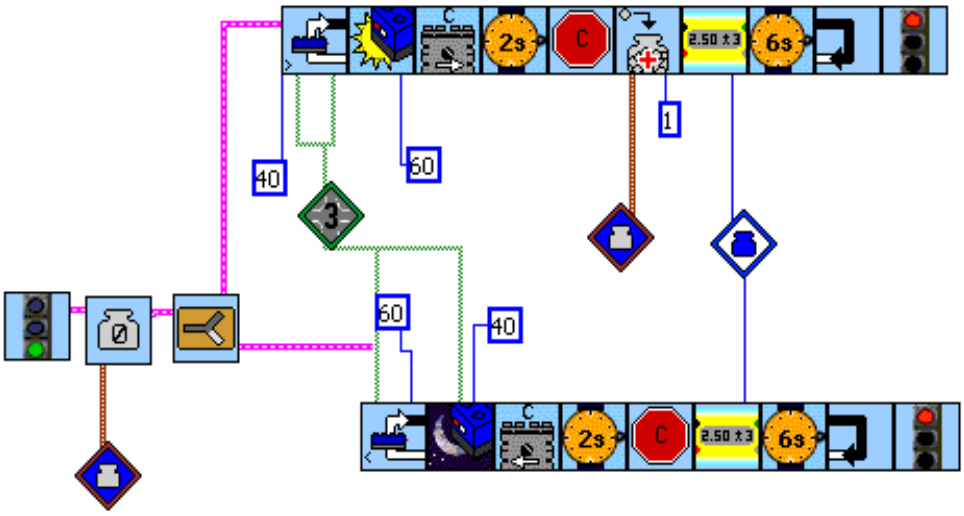



Figure 4

Programming the fridge guy using ROBOLAB™ is slightly more complex than it first seems. A *blue container* is used in my program as a means of counting how many times the bright light is recorded. The RCX LCD only displays the container value for six seconds each time the light changes. This program will run up to fifteen times before ending, this fifteen times is set by default in the loops at the start and end of the program. These *loops* work when the light sensor is greater or less than the wired in percentage. This therefore prevents the programme going constant that would occur if an ordinary *Jump* and *Land* was used. Also there is a tolerance between the two percentage threshold levels this again is used to prevent the program operating by direct daylight (room lighting 48%).

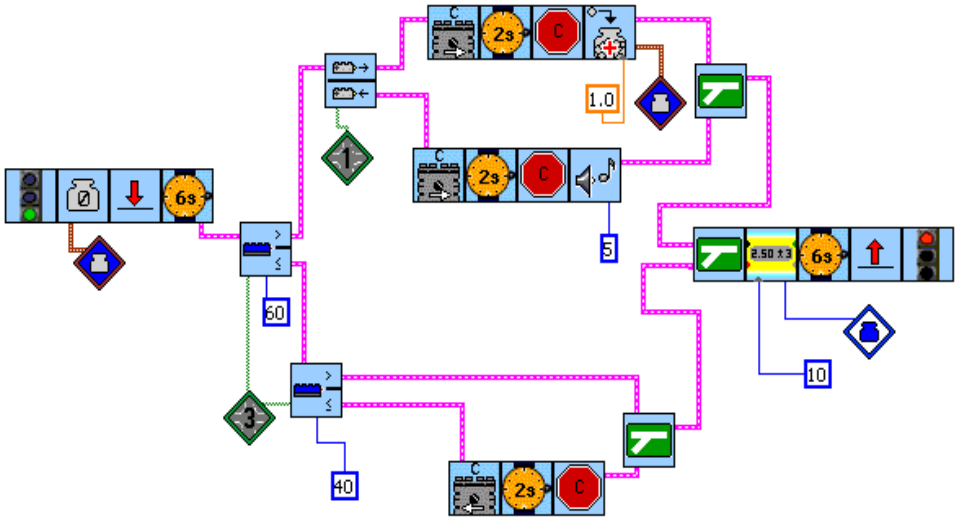



Figure 5

This program shown above satisfies the brief as it allows for all the possibilities. The *Blue container* is set to zero. Then the program waits for 6 seconds this gives you time to hit the sensor and let the light sensor adjust to the percentage of light. The *Light sensor fork* acts as a task split and the program branches of from here.

If the light sensor is reading greater than 60% then the program takes the top route. There again is a *Touch sensor fork* and the different route can be taken depending on the condition that the sensor has read. If for example the sensor is left pushed in the program will turn on C for two seconds and play a sound (5). It then goes to the *Merge Fork* and displays the value of the *Blue container* for six seconds on the LCD. The *Red Jump* then starts the programme back to the start again where the *Red Land* is.

However if the first light sensor is less than or equal to 60% the program takes the lower route to another *Light sensor fork* set to read percentages at 40%. If the light is lower than 40% the motor C reverses rising the glasses and the LCD displays its readings.

It is important to note that the *Blue container* be reset outside the loop as if it was inside the loop after every time it would reset and the readings would be 0 constant. 1.0 is added to the container only when the light is above 60% and when the touch sensor is released.

Notice the need to use a *Fork Merge* when sensor forks are used. They are not like tasks splits that need an individual *Red Light* end. It is also possible to jump over *Sensor Forks* and land on earlier program parts.

## Module 17: Colour Sorter Robot

The challenge is to make and program a robot that dispenses different coloured spherical balls into different areas/cups. You can use ball bearings with electrical tape over them.

1. Use the light sensor to detect the light value of the bearings.
2. Use the treads to drop of the different ball bearings
3. Use a rotating mechanism that allows the sorter arm to rotate and place the different coloured ball bearings into different areas.

You will need to record the value of the different coloured ball bearings for programming the robot. You may use the LCD on the RCX brick to display the light percentage value of each different colour or use the RCX sensor link to the right hand side of the title big block. When programming using the ROBOLAB™ language you can use the same results or find out the values using the investigator-programming environment.

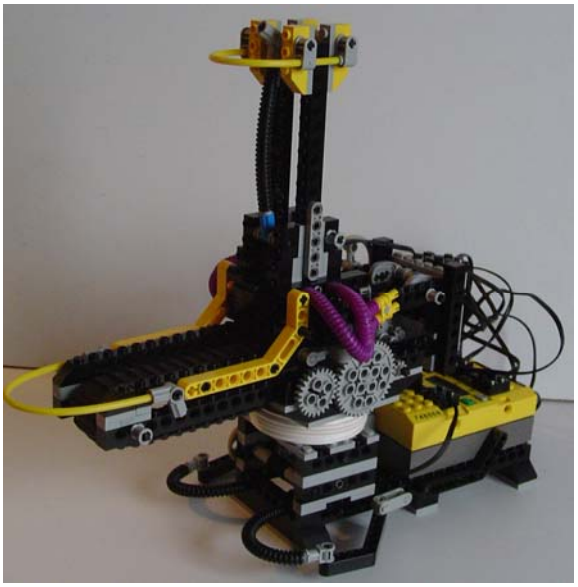


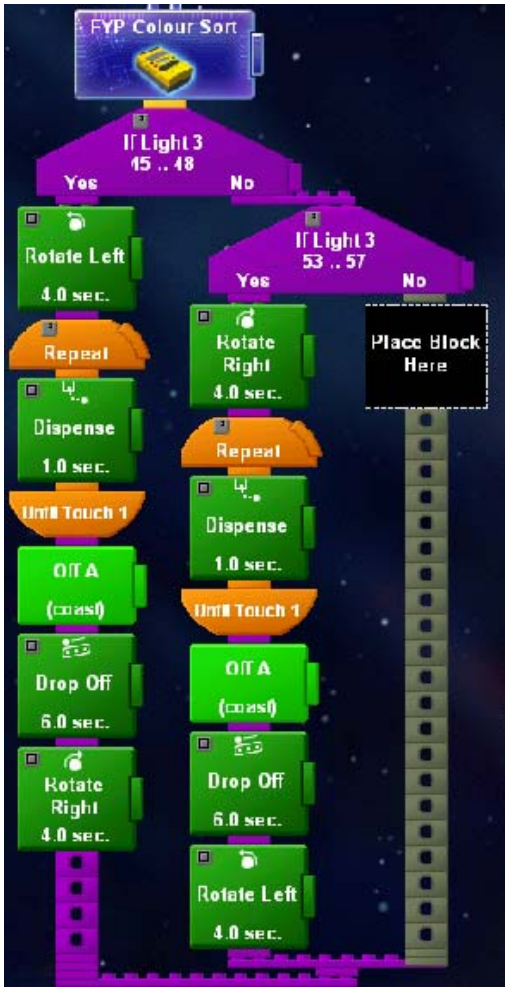
Figure 1

Figure 1 shows a typical example supplied with the RIS™. It is a good idea to look at the video on CD to see what exactly is expected from your robot. If the RIS™ is available to you, you should consider the design that they have prescribed and modify this to have better robot operation. From the video you notice that the robot has being modified to allow the camera to see what is happening. This is not necessary for your design.

To program this robot to perform the task it is a good idea to break the program down to manageable steps and download and test. Modify and then move to the next step. Refer to you flow diagram or logic statement. To make programming easier it is a good idea to write down what device is connected to what port. This also speeds up programming. The RIS™ supplies a sample program that is also on CD however it is more beneficial to design your own. In the following pages I am going to discuss the program's I designed using both languages. It is stressed that you should design your own program first and then consider the sample solution supplied after your robot is operating correctly.

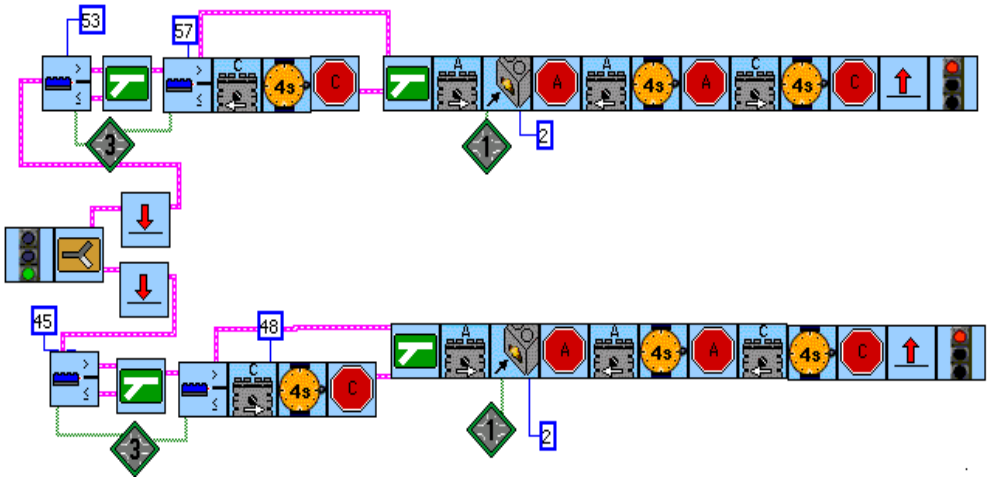
### RCX™ Code

Firstly one needs to find out what light percentage readings the different spheres have. This is achieved by selecting small blocks *Display value Light 3;* this then displays the value on the LCD. It is best to have the sensor in place on your design to avoid any inconsistent readings. This eliminates surround light factors, for example the white table top as a background will have a different reading than the same sphere in a black Lego surround. The program and explanation follows.




The program operates on Yes/No blocks. The light percentages were recorded and used in the program. The white sphere ranges in colour between 45 and 48% if this is true, the Yes option is selected. The arm rotates left and the sphere is dispensed until touch one and then stopped. The treads turn on a drop off the sphere in the correct area and then the arm returns to the starting point.

The other possibility is that the sphere is brown; the only difference here is that the arm rotates in the opposite direction.




Programming the colour sorter using the ROBOLAB™ code is slightly more complex than using the RCX™ code, there are no *Big Blocks* here and the light sensor options operate differently.

The program starts off with a *Task Split*. Then the program reads the value of the light sensor to see what task to perform. When two *Light Sensor Forks* are placed one after another with a *Merge Fork* in between and different light percentages are wired they now check for a range of light between these two numbers. When a number is achieved either 53-57 or 45-48 will the program rotate the arm (motor C). As mentioned earlier forks need to be merged before the end of programme and in this case I placed it in after the arm has completed rotating. The spheres are pushed out using motor A in forward and then when placed in reverse motor A turns the treads on to drop of the sphere. This design feature is most helpful as one motor can carry out two different actions based on motor direction (construction). Once the sphere is dropped of then the arm rotates back to its original starting place.

## Module 18: The Light Activated Obstacle Avoider

Design and make a robot that will avoid objects, walls and continue along any giving path that the robot takes. The two touch sensors must be used and the robot will need a light activated start and finish, for example a blink to start the programme running and a dull light to tell the robot to stop.

The design for your robot should allow for mobility. The programming language is up to your self what one to use as in previous examples I will take both options and analyse how they operate.

My robot can be seen in figure 1 below and the main chassis does not vary from the line follower. The sensor arms are set at an angle to increase the possibility of object detection and touch sensor activation. The light sensor is placed at the rear to prevent it from damage and it was easy to recycle my design from the line follower. A torch is used to activate the program and a black beam is used to terminate it. This can be seen on the Video in the CD.

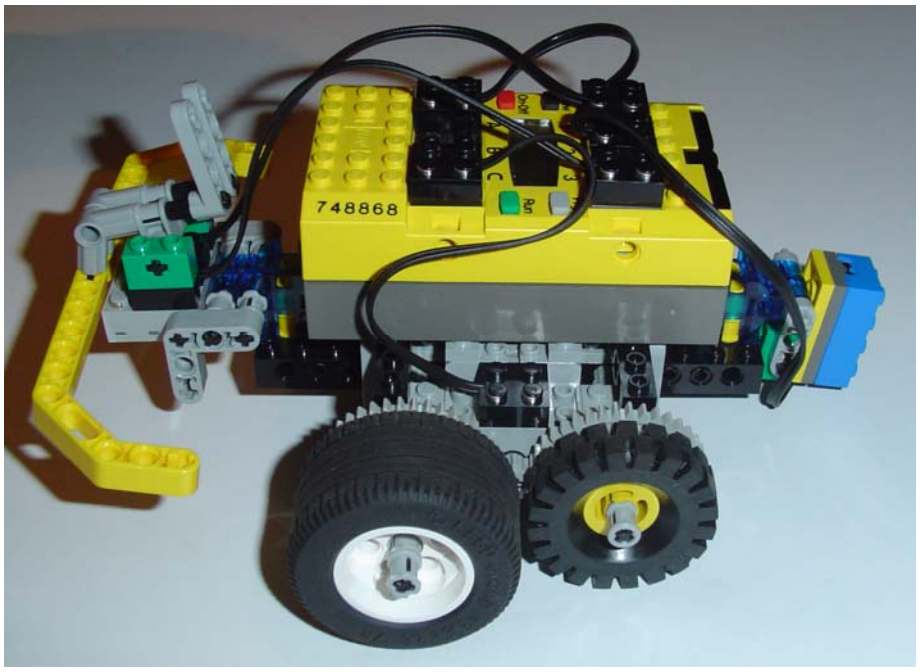



Figure 1

Again it must be stated that it is a good idea to write down a Logical statement and a flow diagram.

*Light activated obstacle avoider.*
*The robot needs locomotion, two arms as feelers touch sensor activated and a light sensor to start and end the programme.*
*The robot needs a means of getting around also*

*Light blink to turn on.*
*Go forward until touch left or right*

*Touch left/right*
*Reverse back and turn away from left/right by turning on the motor opposite it for a second.*
*Then start going straight again, until touch.*
*Turn off by blacking out the sensor.*

Now that you have cleared out what the problem is asking and considered how you are going to program your robot, you should design a program to satisfy the brief.

Try it on your own first and change your program till you get it right. Testing the program is just as important as designing it.

**RCX code**


Figure 2

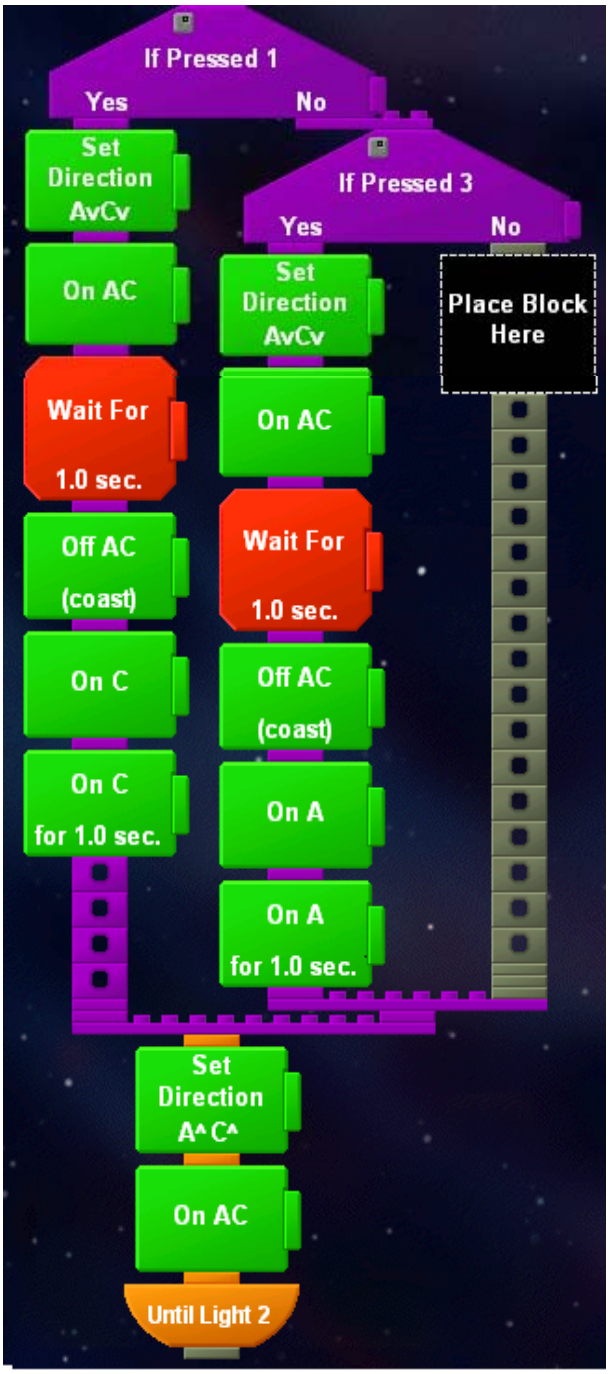
The programme is using sensors to start the programme therefore nothing falls under the main starting block.


Figure 3

The blink option is chosen to start the program. The location of the sensor is at input port 2.

The motors connected to output ports A and C are turned on.

The motors turn off under the following conditions that are under Yes and No function on the following page.




If the touch sensor connected to output port1 is the pressed. Yes
The direction of the motors A and C are reversed then they are turned on for one second. They coast off. And the motor opposite to the touch sensor hit is turned on for one second.
The direction of the motors needs to be reset and they are turned on again.

If the touch sensor at input port 3 is touched then the program skips down to the second *Yes/No* block.
And the program is the same form there apart the opposite motor reverses 1 second more to try and avoid the obstacle.
The program keeps repeating it self until Light 2 that is a dark light being shown over the light sensor.

Figure 4

This is my method of programming the robot to avoid obstacles; however you may have other ideas. One problem with this robot is that when it gets into the corner of a room it takes a long time to get out of it. Can you make any programming alterations to avoid this? (timer-{hint})

**ROBOLAB™ Inventor level 3**

Figure 5 shows my solution to the problem posed by the brief. However the program is missing a part. The program runs well up to the end when you try to end it using the black beam. Due to the size constraints I left it out in my programme. Now it is up to you to program your robot to do this.

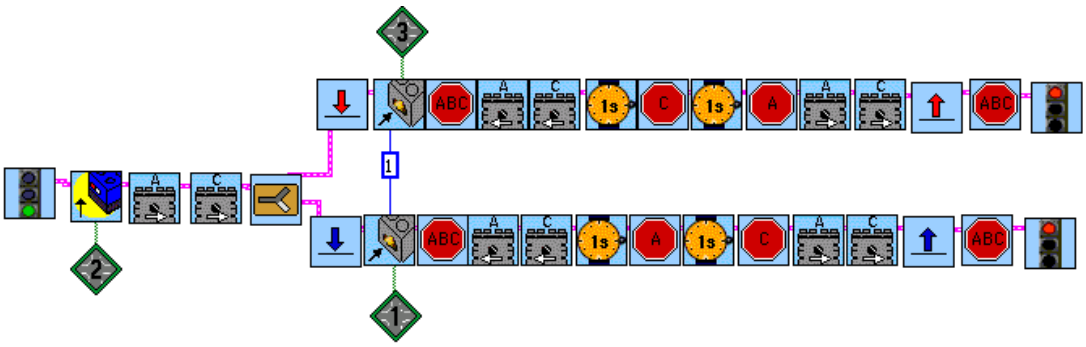



Figure 5

The program has two loops from the structures menu and a task split also. Reverse the motors could be used instead of selecting the motors on the eight separate occasions however it leaves the programming very clear and readable.

You can see the different programming codes go about operating the same task in a very different method. The graphics of ROBOLAB™ code is very helpful and the icon method of program construction in both codes makes it easier to construct programmes.

## Module 19: The Robotic Arm

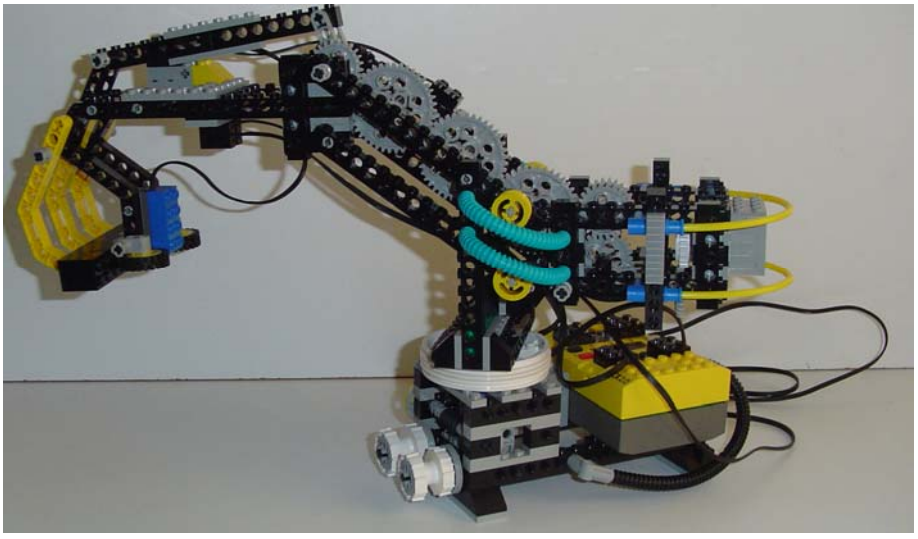
The design challenge is to make and program a robotic arm to pick up Lego beams, rotate and leave them in a desired location.

The light sensor must be used to scan the surface for the object, rotate to the side then lower and open the claw, rotate back over the beam and lift it up. Then the opposite occurs when the beam is placed in its new location.

The Key Features

1. Light Sensor is used to detect when the arm is over the beam
2. Claw opens when lowered and closes before lifting.
3. Rotating base allows the arm to swivel left and right.

The robotic arm can raise, lower and rotate, but it cannot move in or out from the base. Therefore make sure that the beam is placed within reach of the arm. The video on the CD shows the robotic arm working, you can see the Lego beam being moving in after each time the arm moves across to get the light sensor to read it. Eventually after some movement it is located in the right position. Figure 10.1 shows a solution to the robotic arm, now it is up to you to design a working model.



Now that the robotic arm has being designed and made to the requirements. It is time to program it. The RCX code will be taken first and then I will take the ROBOLAB™ programming language.

## RCX Programming

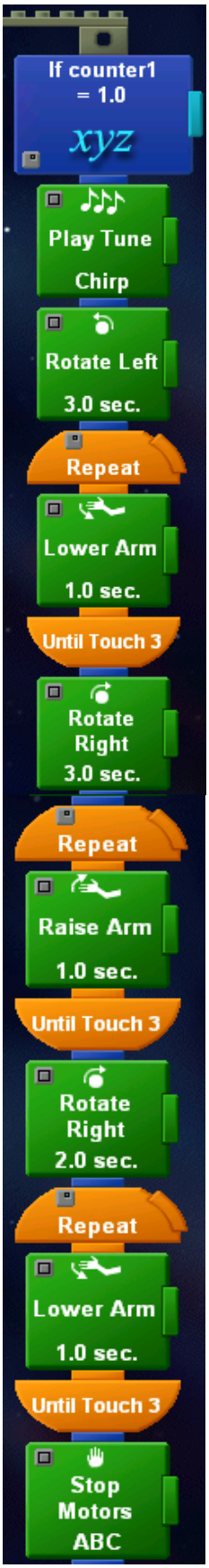

The Robotic Arm

The Arm rotates right for six seconds then rotates left for six seconds. This continues to repeat until the light sensor condition is reached. In this case the light sensor at input port 1 needs to reach a percentage of light lower than 50 %.

When this is reached (black beam on white table) all the output ports are stopped.

If the light received at the sensor is between 48% and 50% the program takes the **Yes** option waits for a second and then adds 1.0 to counter 1.
If it is not equal **No** nothing happens but this will not be the case as the light percentage is predetermined from earlier testing

Figure 2




Only if variable (counter 1) is equal to 1.0

Will this part of the program operate?

The RCX plays notes, this tells you that the sensor has picked up a black beam and is starting the process of lifting it.

The arm rotates left past the beam for 3 seconds.

The arm is then lowered until the touch sensor at input port 3 is pressed.

Now the arm is lowered and the claw is open.

The arm rotates right for 3 seconds to be placed back over the beam.

The arm is lifted closing the claw also and this continues until the touch sensor at input port 3 is pressed.

The arm swings/rotates right for 2 seconds and lowers once more until touched at input port 3.

All motors are then stopped.

Figure 3


The RCX now plays a note at frequency 1000 for 2 seconds.

The beam is still under the arm so it needs to rotate right for two seconds.

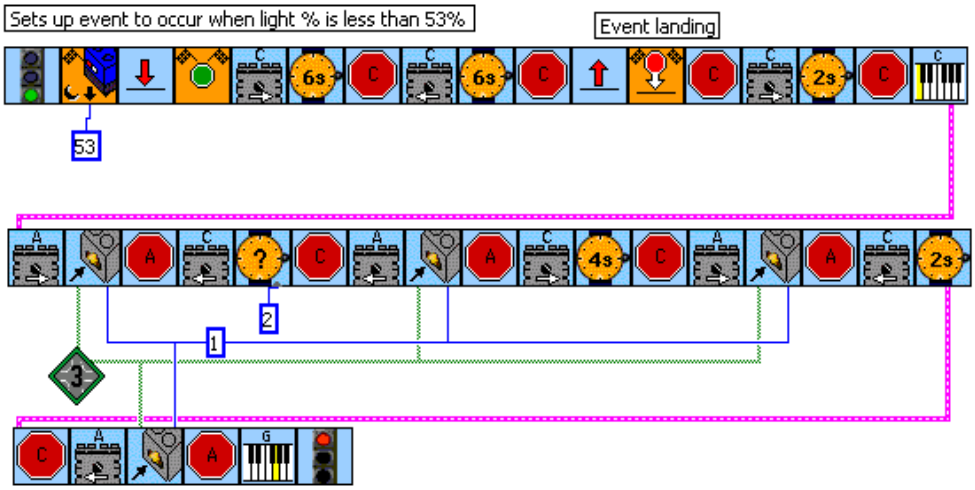
The arm now lifts up until the touch sensor is pressed on input port three.

All motors are stopped and the programme is completed. A small block may be used "Programme End" or the RCX will time out it self after the set time.

Figure 10.4

The above program uses *Big Blocks*, *Small Blocks*, a *Yes or No Variable* based on a light sensor, *Repeat* functions based on a touch sensor and a variable *sensor watcher* that works a stack of commands when a set condition has being met (counter 1 at 1.0 as in the above case). The music or notes are a good way to debug a program and find out where the actual problems are. They can be used as warning sounds for example when the arm is going to rotate also.

## ROBOLAB™ programming




The program above uses *Events* as a method of jumping out of the *Red Jump* and *Land* command.

Firstly the *Dark Event* is set up to be activated when the sensor reads a light value lower than 53% (pre test recorded not the same as the RCX percentage). Then the *Red land* is used to keep repeating the piece of program to where the *Red Jump* is. This rotates the arm left and right for 6 seconds in each direction. Just after the *Red Land* icon the program starts monitoring for the event. If the light percentage is above 53% the program between the *Red Jump* and *Land* keeps repeating. However when the sensor detects a black beam the light % will drop to below 53%. This will satisfy the condition preset in the program and will move the program out of the *Red Jump* and *Land Loop* to the *Event Landing* further on in the program. Now the program has detected the beam and rotates past it for two seconds. A note (C) is played and the arm is lowered until the touch sensor is pressed. The arm rotates over the black beam and lifts it up. The rest of the program is self-explanatory and caries out the same sequence of work as the RCX code.

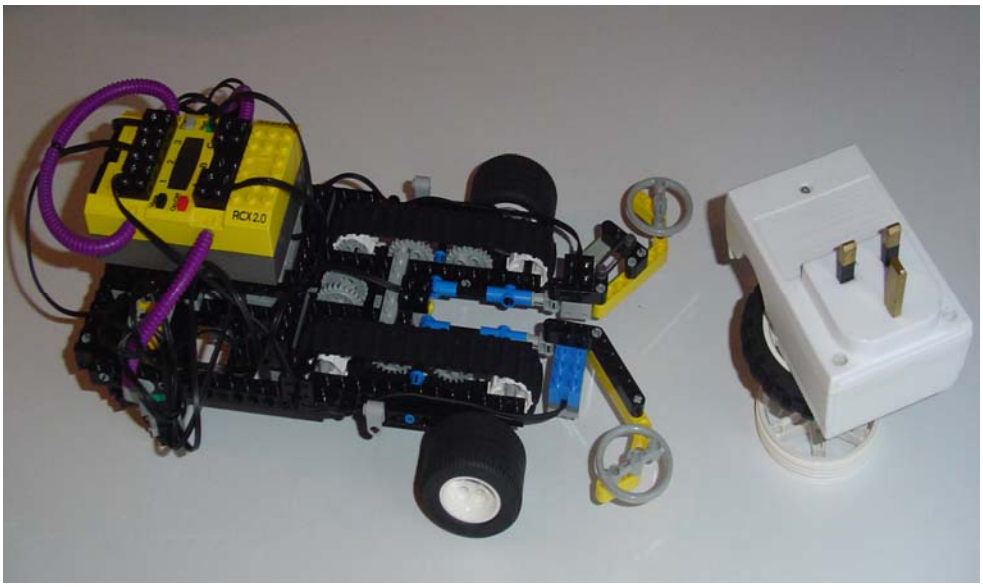
## Module 20: The Delivery Robot

The challenge is to design and program a robot to pick up an object and drop it at a new location. The robot must use motors, touch sensors and the one light sensor.

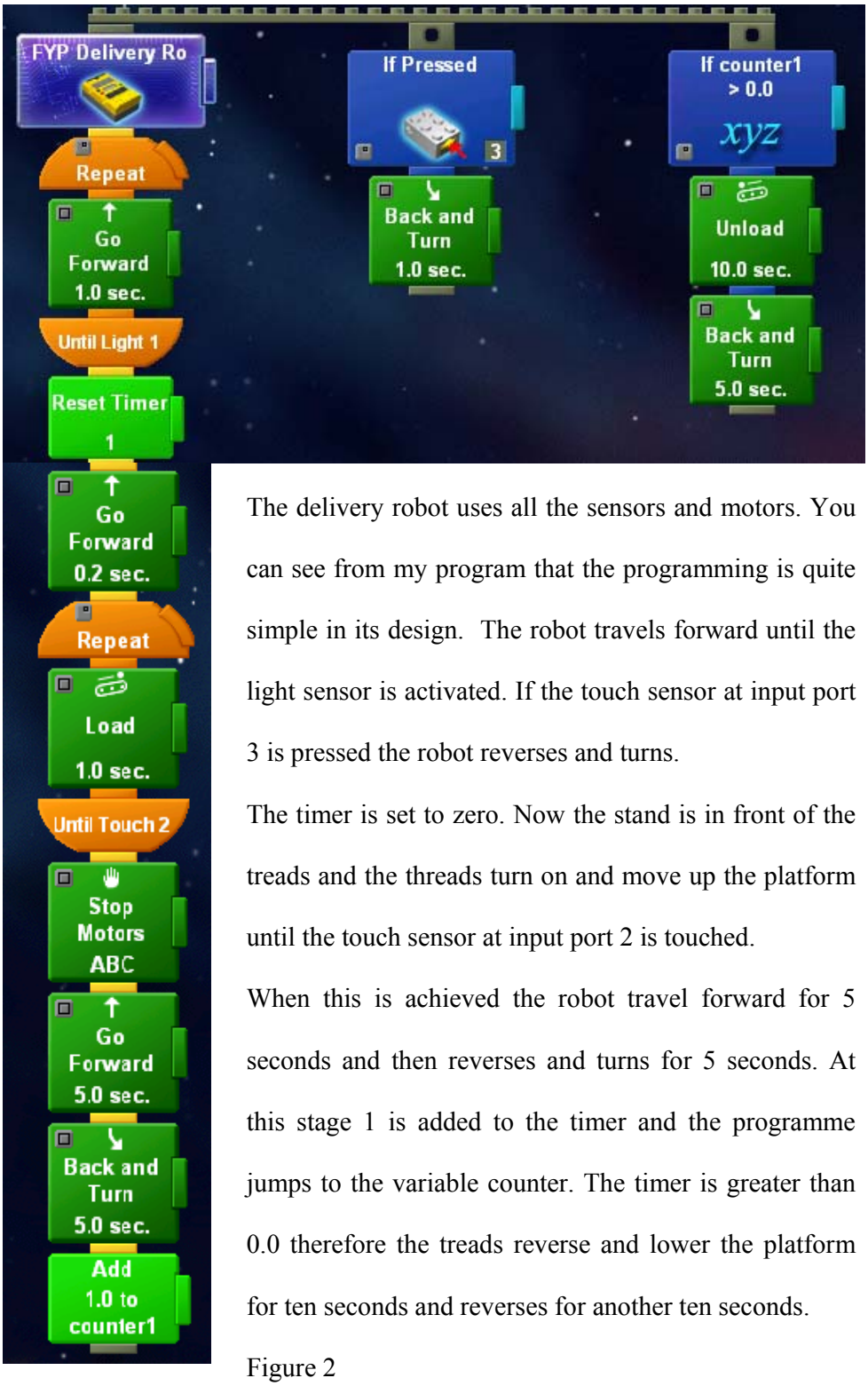
The RIS™ provides you with Key Features that your robot must have a do.

1. Place the object that you want to move on a platform.
2. The bumper has a touch sensor that tells when it runs into the platform.
3. The light sensor is used to tell when the platform is in position for lifting.
4. Use the threads to pick up the platform and drop it off when it is moved.
5. The inner touch sensor knows when the platform is held securely in place.

The robot design is now constrained from the above features it must contain. It's your challenge to design a robot to contain these constraints. My robot shown below in figure 1 is built from the RIS™ interactive screen (Key steps). The programs are designed to meet the requirements of the brief.



**RCX programme**




The delivery robot uses all the sensors and motors. You can see from my program that the programming is quite simple in its design. The robot travels forward until the light sensor is activated. If the touch sensor at input port 3 is pressed the robot reverses and turns.

The timer is set to zero. Now the stand is in front of the treads and the threads turn on and move up the platform until the touch sensor at input port 2 is touched.

When this is achieved the robot travel forward for 5 seconds and then reverses and turns for 5 seconds. At this stage 1 is added to the timer and the programme jumps to the variable counter. The timer is greater than 0.0 therefore the treads reverse and lower the platform for ten seconds and reverses for another ten seconds.

Figure 2

**ROBOLAB**

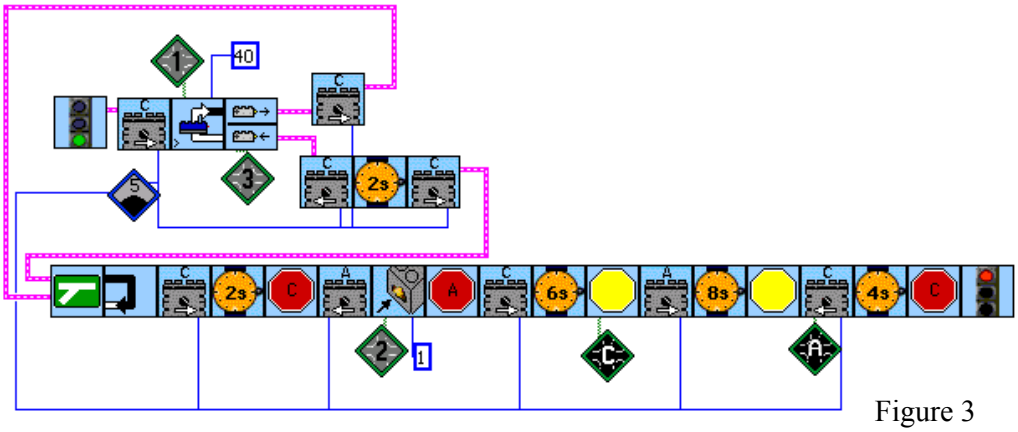



Figure 3

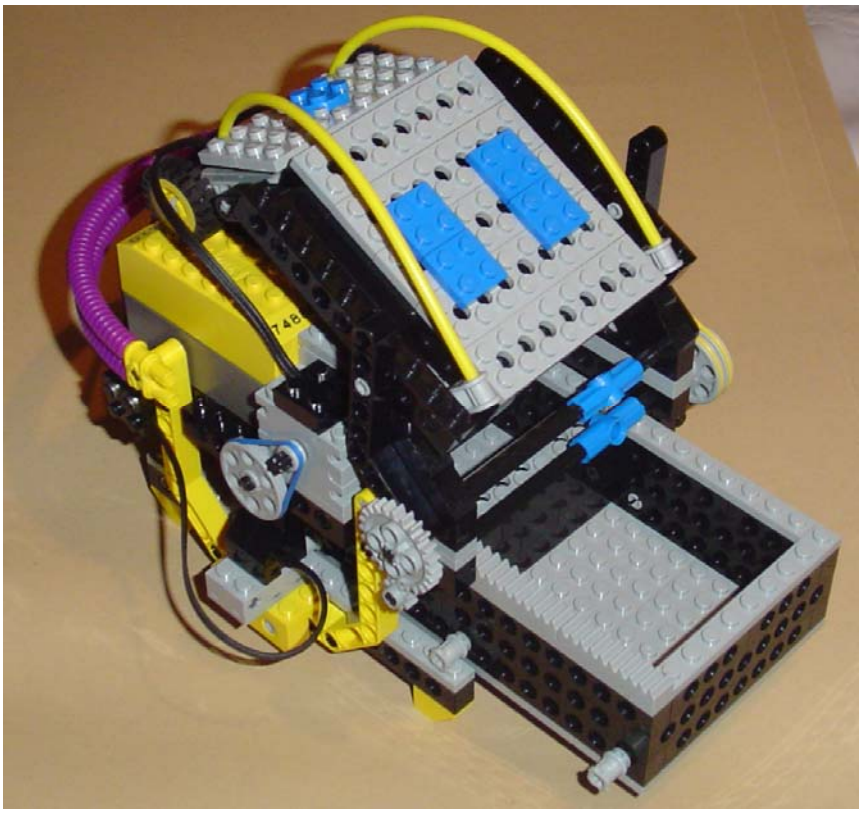
Programming the robot to carry out the same task is quite different than the RCX language. You can see that that motor C goes forward at full speed then it meets a *Loop While Light Sensor is Greater than* wired in variable 40%. This then meets a *Touch sensor Fork* that lets the robot get in line for the platform. The different options are quite simple to read.

The *Merge Fork* needs to be placed before the *End of Loop* in order for the *Loop* to work correctly. The program then continues on from here with the treads rising up the platform until the touch sensor at input port 2 is activated. All motors are stopped. Now that the platform is lifted the robot drives forward for 6 seconds. Then the platform is lowered to the ground for 8 seconds. Notice that the motors are *Floated* for these operations. Then the robot reverses for 4 seconds. To get the program to continuously repeat *Jump* and *Lands* may be used.

## Module 21: Security Vault

The challenge set is to create a robot to safeguard your valuables. Create a security card that can be read by the light sensor (black and white). When the correct security code is passed over the light sensor the door opens and the tray slides out. Also there is a secret compartment that you can open using a different security card.

The interactive program has key steps and 3D views these may be used to build the robot. When built it may look like figure 1.



Now that the security vault is made you will need to program it to do what the design brief states. We will first look at the RIS™ method of programming and them ROBOLAB™. As you might have noted the RIS™ has a sample program

all ready installed to run the security vault. We are going to study this program and see how they programmed it to perform the tasks. The program is broken down into manageable sections and is explained through out. It is an idea to have the program open, and follow on screen. Left to right, top down sequence is used.

**RCX Code**


The name of the project appears as the name you save it as.

Your L.C.D. screen now shows how many lines pass the light sensor.

The white card is set to Light 3 (Input port 3)

5 is added to the white variable.

The black lines are also set to Light 3

2 is taken away from the Darkline.

For the front to open 3 black lines need to register to open

For the hidden vault to open 5 black lines need to pass the light sensor

This is the security card now set up so now we need to differentiate between the different light types and what to do when the sensor detects these.


The light sensor is set to detect a range of light between the white card (bright) and 100 on the scale.

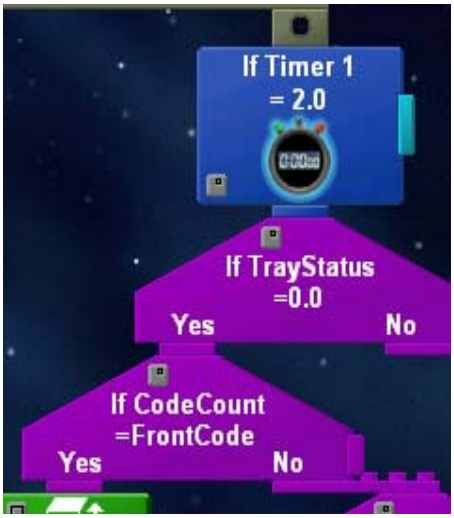
If this occurs the timer 1 is reset to 0

Figure 3


The light sensor is also set to detect Darkline, only if the line is between 0 and Darkline.

Connected to this is a *Yes* or *No* block. Variable is selected and is set equal to 0.0. If yes nothing happens and skips over but if it is not equal to 0.0, 1 is added to the code count.

Figure 4




When the timer 1 reaches 2

Yes or No function with variable equals to 0.0

We are going to take the yes option and then come back and do the no option later.

Figure 5


If the code count is equal to the front code then the following must happen.

The door opens for one second

The tray comes out for .5 of a second to clear the touch sensor.

It will continue to come out until the Lego pegs hits the touch sensor

Touch sensor touched

All out puts are stopped.

Tray is out therefore set the status to 1 (telling the program that it is out)

Reset or set the code count to 0.0 to start the process again.

Figure 6

If code count is equal to the front code is programmed above (Yes), however if it is no then the No function is used. The next page details this.

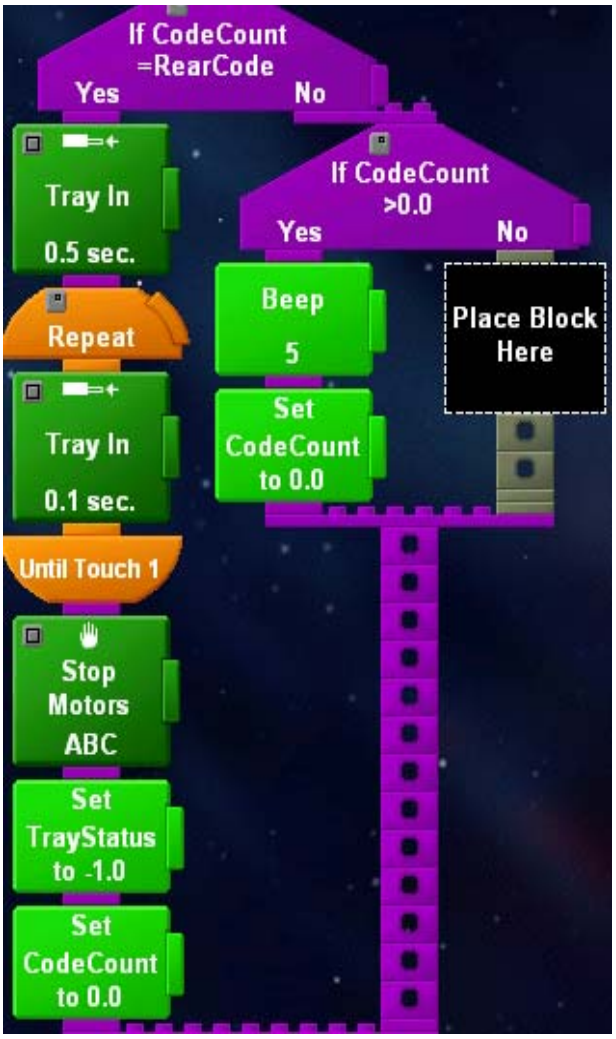



Figure 7

If the code count does not equal the front code it might equal the rear code count **Yes.**

If this is the case the tray will reverse (in) until the touch sensor is activated. All output ports are then stopped.
The tray status is set at –1 to tell the program that the tray is out at the rear.
The code count is then set to 0.0.

However if the code count is not reached **No** (i.e.) wrong security card is used, then the RCX will beep and the code counter will be reset


Figure 8

From figure 4.5 the NO function for (if tray status = 0.0)

If the tray status is > than 0 **Yes**.

The tray moves in until the sensor is activated.

Then all the output ports are stopped.

The door closes for 1 second and the tray status is set to 0.0 due to it being back in the vault.

The code count is then reset.

However if the tray status is not > 0.0 it must be –1 (**NO**) Therefore the secret compartment needs to be closed by moving the tray in the out direction until the touch. Then the tray and code needs to be set to 0.0.

As can be seen from the above programming we do not deal with specifics or individual motors. The big blocks are all ready supplied for us in the program so just imagine if we had to go through the programming using small blocks. Now that we know and understand this programming, we should test it out to see are their any bugs. The next section we are going to program the vault to do the exact same using ROBOLAB™.

**ROBOLAB™ programming**

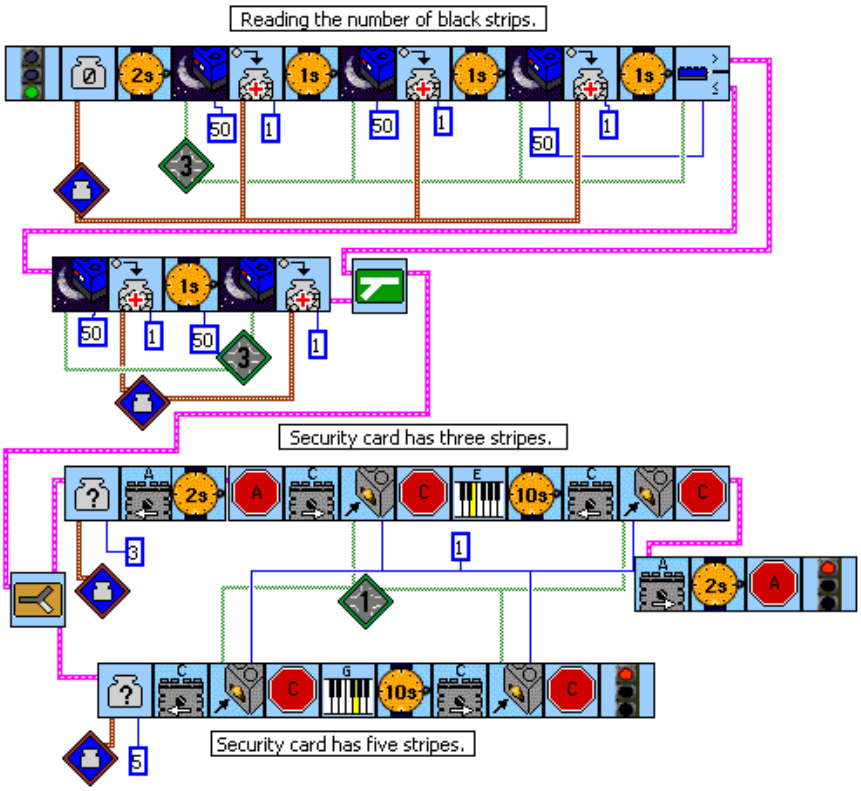



Figure 9

The program above is quite complex so I will break it down into rows.

The first resets a container (blue) and *Waits* for two seconds.

Then the light sensor is activated and looks for light less than 50%. If this is true 1 is added to the blue container (each time), this continues two more times and reaches a *Light Sensor Fork* if there are more strips the lower option is taken, but if there are only three strips the top option is taken and this is *Wired* straight to the *Fork Merge.*

The second row is a partial copy of row 1. This counts the next two strips on the security card.

A *Task Split* is used for the different security cards.

Row number three is programmed to be activated when the *Blue Container* is equal to three. This opens the front door and slides the tray out for 10 seconds and then the reverse happens after 10 seconds has elapsed and the warning sound.

The final row is only used when the blue container is equal to five (security card has 5 strips). The tray slides out the rear until the touch sensor is pressed plays a warning note and slides back in after 10 seconds.

Reading the number of black strips is a variable on time delays of 1 second this may be reduced if need occurs.

This program is more cumbersome than complex. The EDIT, Copy and Paste additions should be used here. To get this program to record how many black strips have passed the light sensor takes a lot of programming and debugging. The containers are invaluable here and always remember to reset them at the start of every program as the container value is held in memory by the RCX.

## Module 22: The Brick Sorter

The challenge is to design and program a robot that will sort out 1 x 2 yellow and black Lego bricks.

The constraints of the brief are as follows:

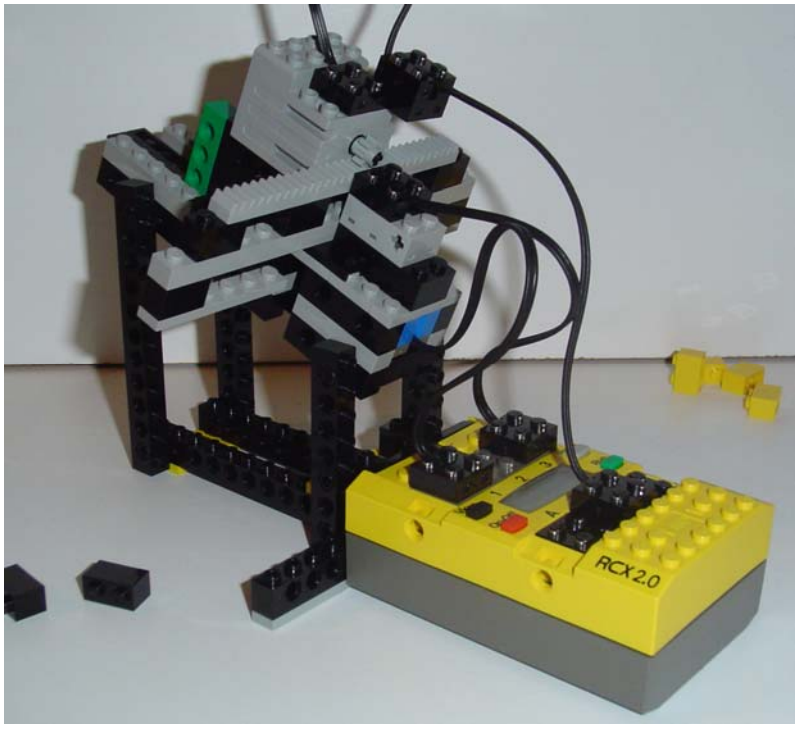
1. The robot should have only one motor, one touch sensor and one light sensor.
2. The Lego bricks must be feed automatically (gravity!).
3. The robot should be capable of separating 8 bricks in one pass.
4. The program in RCX must not use *Light Sensor Watchers* and in ROBOLAB™ the program must use *subroutines* and *Task Splits* should be avoided.
5. It will be important to use time delays in your programming to let the robot process information. Motor power is best set low.

It is now up to you to design and test your own brick sorter. The programming should be carried out taking your design solution into consideration. Pre-tests are important to find the light value of the different colour bricks. The program for these tests should also be saved. Remember it is important to carry out these tests in the same environment to where the robot will be operating (light percentage variables).

There is a sample solution supplied and discussed over leaf however one should design, program and test your own solution first.

**Sample solution:**

The design meets the requirements and the bricks are gravity feed into the robot. The motor works the dispenser arm and the direction is programmed according to what percentage light is recorded. The design is shown below.




The touch sensor is used to tell the RCX where the dispenser arm is. The mechanism used to transfer rotary motion into linear motion is the rack and pinion. It is important to note that the gravity slide for the 2 x 1 bricks are long Lego beams turned upside down to allow the bricks to slide.

Programming the robot to carry out the separation will only involve three pieces of hardware, motor @ B, light sensor @ 1 with the touch sensor @ 3.

My sample programmes will be discussed also taking into consideration other options that do not work and the reasons why.

## RCX language

The method of programming this robot to perform the task is quite important. As mentioned in the constraints *Light Sensor Watchers* should not be used, the reason behind this can be demonstrated by programming your robot using them. The RCX checks the sensor inputs frequently therefore if the programme is pushing out a yellow brick and the light sensor picks up another brick the program jumps back to the start and may even jump form one stack controller to another from black (38-42%) to yellow (54-56%) in our case. Figure 11.2 shows the case that should be avoided.


This is called a hardware conflict however if two light sensors were being used with one for indicating the black bricks and another for indicating the yellow bricks the program would be successful. However due to our limit of only using one sensor and considering the above information the only other solution is to use a *Yes or No* command. These commands are versatile as they can be doubled up and act as sensor testers and only carry out the commands when the set condition is achieved. Figure 3 shows the starting point for the program.

Figure 3


The brick sorter programme continues from the previous page. The RCX Beeps when the black brick is placed in front of the sensor. The power setting is lowered this is a condition set by the brief. The direction is *Reversed* and this continues *Until Touch 3*.

It is important to note that this repeat option is set to operate when the sensor is released. The sensor is released when the program starts due to the design of the dispenser arm but the computer does not recognise this so the sensor needs to be pushed in and released to turn the motor off.

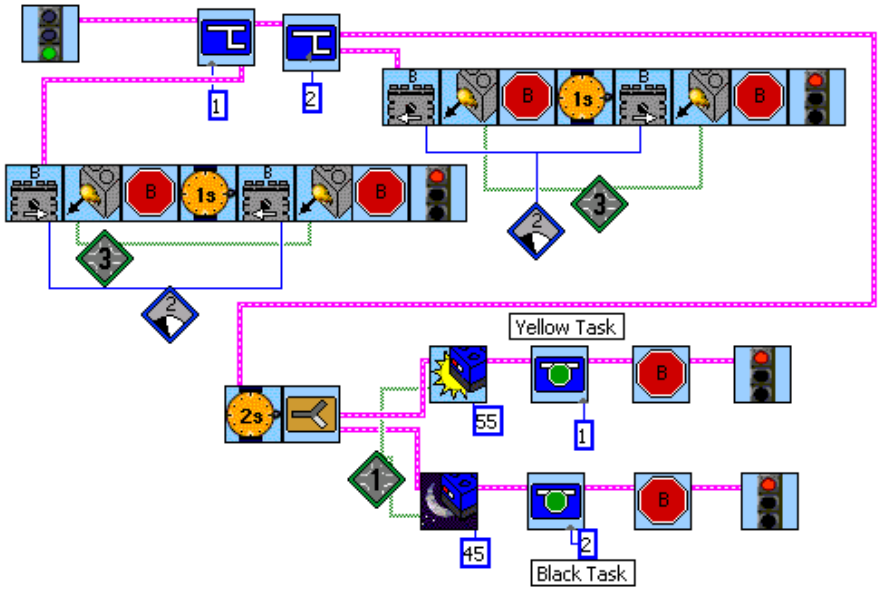
The program waits for 1 second then reverses until the touch sensor is released. The program again waits.

This repeats forever so it will be capable of sorting out 8 bricks in one pass.

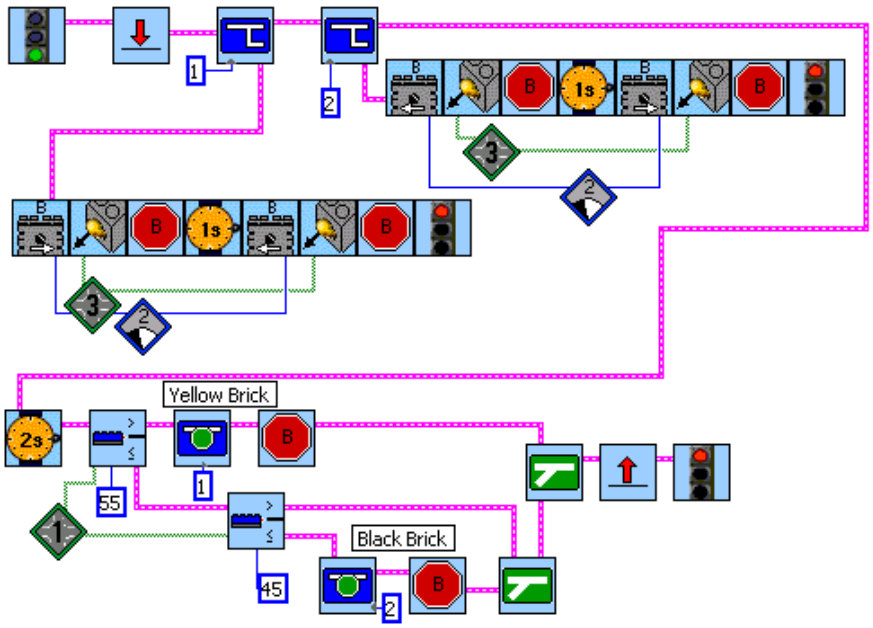
This program will follow through regardless of the next brick unlike the *Light Sensor Watchers* program.

## ROBOLAB™ programming

Programming the brick sorter using ROBLAB™ offers a range of different methods. The design brief states that the program uses *Subroutines*. I have made two *Subroutines* one for the black brick and another for the yellow brick. My first program used a *Task Split* to differentiate between the bricks and then the *Subroutine* would be run. This proved to be unsuccessful, as the robot was not functioning correctly. After considerable testing and debugging my programmes I changed the *Task Split* and used two *Light Sensor Forks* instead. The *Task Split* program was acting similar to the *Light Sensor Watchers* in the RCX code. Hardware conflicts and jumping between tasks was the problem here. For example a black brick is being dispensed and a yellow brick behind it then comes into view of the light sensor then program jumps to the yellow task even though it may not have completed the black brick task. The light sensor readings may be different between the program languages; this is why it is important to test the readings before programming. Figure 5 shows the program with a task split that does not work.



This program shown below in figure 6 substitutes the *Task Split* for a *Light Sensor Fork*. The first *Light Sensor Fork* has 55% wired to its terminal, any light greater than this signifies that a yellow brick is present and the less than or equal to terminal is wired to another *Light Sensor Fork* with a light percentage of 45, that of a black brick. The arrangement of the *Fork Merges* is not important as long as they are used. This program will not jump from one *Fork* to another. This program will repeat forever due to the *Red Jump* and *Lands* in the construction. The subroutines are used and the program satisfies the design brief.



## Module 22: Task sheet

1. Card playing is beneficial to the person dealing the cards, due to a myth that they know the cards. To elevate this problem design and make a playing card dispenser that can rotate and dispense cards. The brief is to dispense 5 cards in three bundles. The first throw gives each player 2 cards the second throw another 2 cards and then 1 card per bundle.

2. Design and make a model elevator that will move Lego block up a certain distance and then move down when activated to do so. All sensor components should be used to solve this problem.

3. A robot is required to climb chicken wire. The robot should know when it gets to the top and stop climbing. Once it reaches the top of the chicken wire frame it returns back down the wire to the ground.

4. A cleaning robot that does not go out side a perimeter is to be designed and made. The robot should avoid obstacles and continuously clean the surface.
5. Design and make a device that will follow a torch light. When/if the device can not find the light it should start searching for it after 5 seconds idle.

6. Design and make a racing car that will travel at the highest possible speed using the touch sensors as revs/gearing up and down. The design should reflect a F1 racing car.

1. Candle Snuffing

**Challenge description**

The challenge is to build and program an autonomous robot that locates and extinguishes the flame of a candle within a given space and time.

(Caution: use a flash light instead of a candle)

**Playing field specification**

Undefined, Contains a lit candle, the flame of which must be of at least several centimetres tall.

**Rules and scoring**

Robots must operate autonomously. They must extinguish the flame correctly through snuffing it out. The robot that snuffs out the flame in the fastest time wins.

Source: BEAM Robotics, Mark Tilden, University of Waterloo, Canada.

2. Robo Hockey

**Challenge description**

The challenge is to build and program a robot that pushes bottle caps from the centre of the field to the robots goal line.

**Playing field specification**

A square shallow plywood box. The outside dimensions are 122 cm in both length and width. The side's are14 cm high. The bottom should be painted white and this is to allow for the bottle caps to be seen.

**Rules and Scoring**

Each robot competes in turn in a round. Each round lasts for two minutes. The robot which pushes the most balls through its goal in that time wins.

Source: John Bilotta, Rhode Island

3. Aquavore

**Challenge description**

The challenge is to build and program an autonomous swimming or rowing robot. (Caution: do not use the water in the fish tank)

**Playing field specification**

The playing field is a water tank with about 15 cm water depth.

**Rules and scoring**

The robot must swim, crawl, jump, fly or row over to reach the finish line. The robot must fit within a 150 mm cube.

Source: BEAM Robotics, Mark Tilden, University of Waterloo, Canada

4. Aerobot

**Challenge description**

The challenge is to build and program an autonomous robot that flies and drops a payload on a target.

**Playing field specification**

A square zone with a target placed on the floor.

**Rules and scoring**

The robot flies and drops its payload on the target, than return to its launch pad. Points are givin for control, accuracy of the pay load on the target, repeatability and time.

Source: BEAM Robotics, Mark Tilden, University of Waterloo, Canada

5. Botball

**Challenge description**

The challenge is to build and program an autonomous robot that moves "rock" samples from the playing field into a specific goal area. The robot must compete against another robot during each round.

**Playing field specification**

The rectangular area is divided into two designated areas with a goal at either end. With rocks that are randomly placed.

**Rules and scoring**

Two robots compete with each other in 2 minute rounds. The robot with the most rocks in its goal zone at the end of the round wins.

Source: Unknown

6. Basketball

**Challenge**

The challenge is to build and programme an autonomous robot that collects a ball from a specified area and then moves that ball to the basket and drops the ball in the basket. Two- four robots competing at the one time works best.

**Playing field**

The same playing field as number 2 can be used. A white tag or marker is placed beneath the ball dispenser to trigger sensors.

**Rules and scoring**

2 minute rounds. Students hand feed their robots the balls at a pre designated area. The robot with most balls in the basket wins.

Source: Janice Borland, Texas.

7. Robo Ping Pong

**Challenge**

The challenge is to build and program an autonomous robot that moves from the end of a specified field to the centre where a line of balls is placed. The robot must compete against another robot to push as many balls as possible towards the opponent's side of the field.

**Playing field specification**

The same field as number 2 can be used. The base is tilted with the bottom half in black paper, therefore you will need to let your programme know where it is in relation to the base.

**Rules and scoring**

Two robots compete in 1 minute rounds. The robot with the fewer balls on its side of the court wins. Play until one robot in the class is decided winner.

Source: Fred Martin, MIT

8. Legged robots

**Challenge**

The challenge is to build and program a robot with legs (instead of wheels) that can compete in a race with difficult terrain.

**Playing field specification**

Legged autonomous robots face off against each other in distance/progress/ability challenges over various rough but equal terrains.

**Rules and scoring**

Robots are compared on a ratio of size, number of limbs ,lift/drive capability, dynamic versus static functionality and terrain-handling ability. Robots are awarded points based upon their ability to handle the broadest range of challenges. The competitor with the most capability wins. The contest is open to various designs.

Source: BEAM Robotics, Mark Tilden, University of Waterloo, Canada

9. Trailblazers

**Challenge**

A robot or team of robots, blaze a trail from a start position to a goal with a variety of obstacles. Robots have a choice of following a line to the goal or blazing a path through an unknown territory with obstacles.

**Playing field specification**

The same field as number 2 can be used.

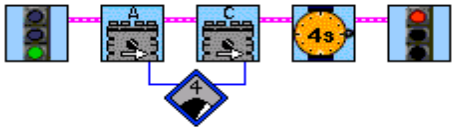
**Rules and scoring**

Each robot must fit within a 150mm cube at the start of the match. After starting the robot can assume any size no larger than 350mm. A robot can bump into walls without penalty. A robot can not climb over the obstacles. The shortest time of the two match scores is the final score. Any robot capable of taking the can back to the starting area will have its time reduced.

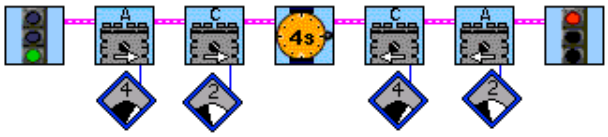
Source: Prof. Bob Avanzato, Penn State Abington College.

## Troubleshooting with ROBOLAB

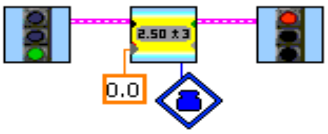
1. Motors A and C are programmed to turn on for 4 seconds at power level 4. This program is never ending, the program should stop after 4 seconds can you suggest any methods for solving this problem.



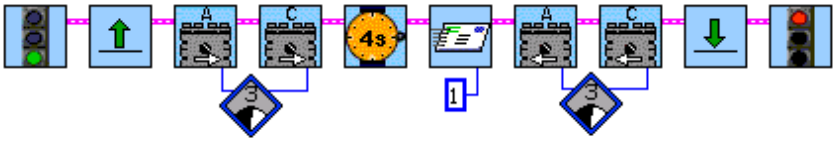

2. A robot car is required to turn going forward and then after 4 seconds return to its original starting point. Correct the program and test to see are you correct.



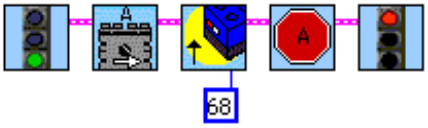

3. The liquid crystal display screen is required to display the value of the blue container, however it does not in this program, resolve the program error and test to see does it work.



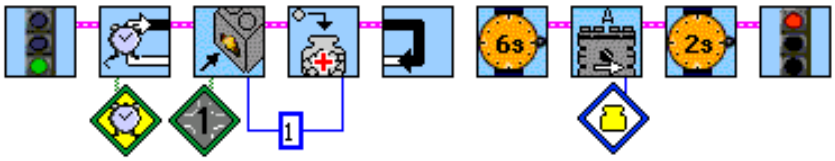
4. A signal tower is required to lift the RCX up for 4 seconds and send mail to another RCX then return to ground level and repeat the process. However this is not happening, solve the problem and test you solution.



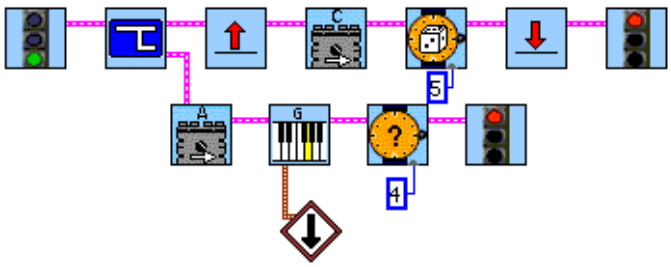

5. An automatic fan is not working as when the light goes higher than 68% the motor does not turn of. The light sensor is connected to input port 3 and the motor is at output port A.



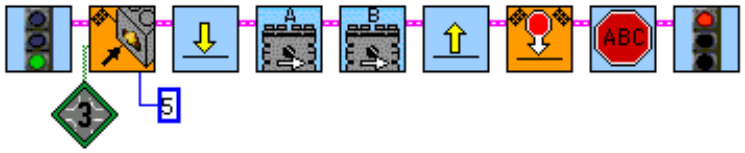

6. The A motor is required to turn on at a preset power level determined by the number of touches on input port 1. The program waits for 6 seconds and then caries out the motor at the speed set. However this does not seem to be working. Correct or debug the program to get this working correctly.



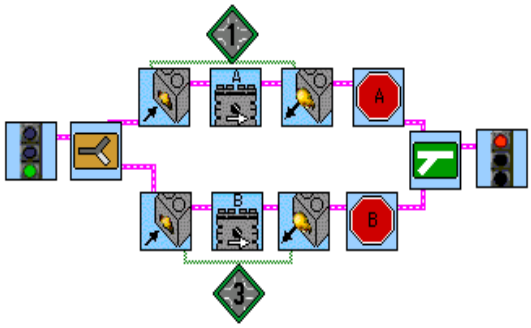
7. Debug this program so that the subroutine starts after the random number of seconds is elapsed. This program should continuously run forever, but would it any way? Try it out.




8. The program below is not working according to as it should. When the sensor is touched the motors should stop however they do not and just keep going.




9. This program was designed by a person not used to ROBOLAB™ programming; it was there solution to the remote control, debug the program and why does it not work.



## Module 23: Investigator

The ROBOLAB™ software has another programming environment called Investigator. This programme option is selected by clicking on the Investigator icon on the out line screen and is a different environment to that of both Pilot and Inventor. Investigator is a program used for data logging; the RCX can be used as a data-recording device. The RCX is a Programmable Interface Controller (PIC). Up to recently controllers and data loggers were separate devices and a system that contained both functions was regarded as a weak system and not a good buy. However today the Lego® Mindstorms™ set does both with incredible power.

Control and datalogging

The RCX can perform both tasks together in the one program. But I will deal with them separate to make compression easier.

### Control

The program that is downloaded into the desired program slot controls the RCX. Once this is activated when the run (green) button is selected the RCX will start the program. The program can be designed to allow for multitasking and can have up to three inputs and three out puts.

## Data logging

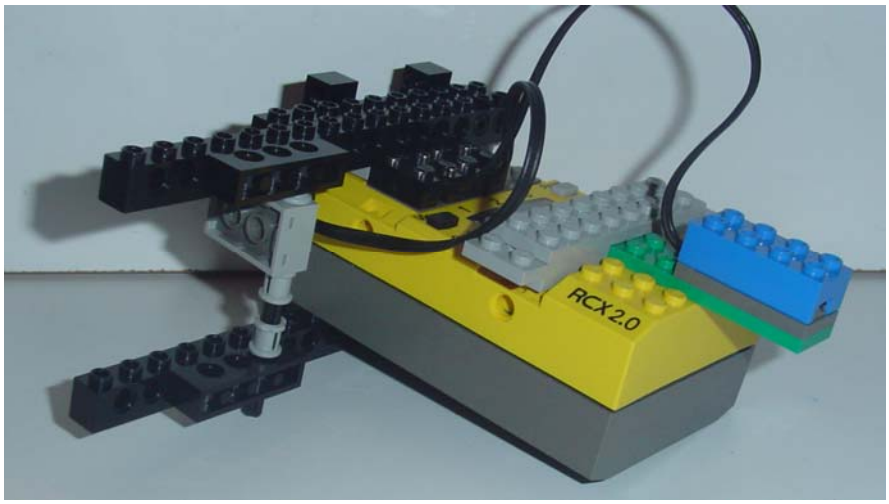
Data logging is the recording of data. Usually this data is too hard for a human to record for example a device may emit a signal at a frequency of 100 times a second, or in the other extreme might only emit a signal every 12 hours. The RCX is programmed using the PC and when the data is recorded it is uploaded to the computer and the data is analysed. The advantage of this data logging device it that is does not need a computer to record its data and is very mobile. The memory is quite large and the RCX will time out before the data logging memory is full.

## Data logging using the RCX

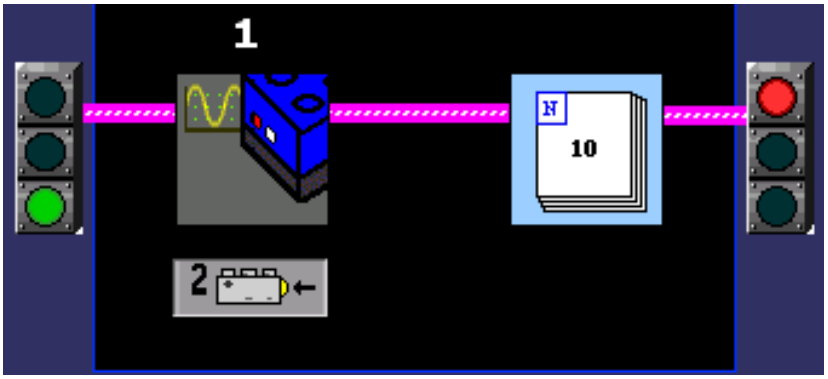
The Investigator level should be opened now. The first design brief that I will talk you through is:

Record the light intensity as it increases the closer you get to a sun kissed window. The light sensor must only take a reading of light intensity when the touch sensor is touched. The maximum number of points to be taken is ten; therefore take ten steps back from the window and click the sensor each step to the window.

The design brief solves the problem itself but now we need to design a hand held RCX with a light sensor and a touch sensor. The reflection of the hand should not interfere with the readings of the light sensor. My design is shown below. It is similar to a camera design no fingers in the way of the shutters.



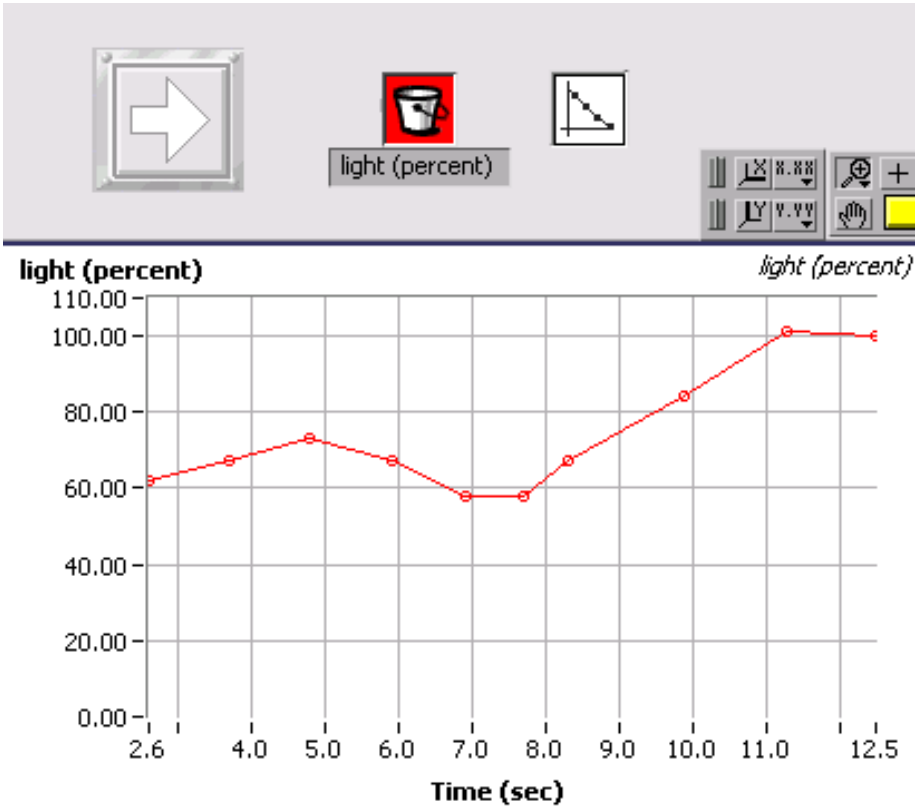

The programme is designed in programme level 1.




This programming is similar to that of the Inventor pilot levels. The sensor records the light values when the touch sensor is pressed. To programme the RCX in the Investigator level the highlighted area should be selected and this gives you a selection of programming level (1-5).


Once the program is downloaded you can carry out your data logging experiment. When the data is collected you will need to up load this data to your computer this involves selecting the up load data on the main menu that is directly below the traffic light icon in the menu shown above. On this screen is an icon similar to the download icon that up loads the data. The IR tower should be near by to cater for the data up load.

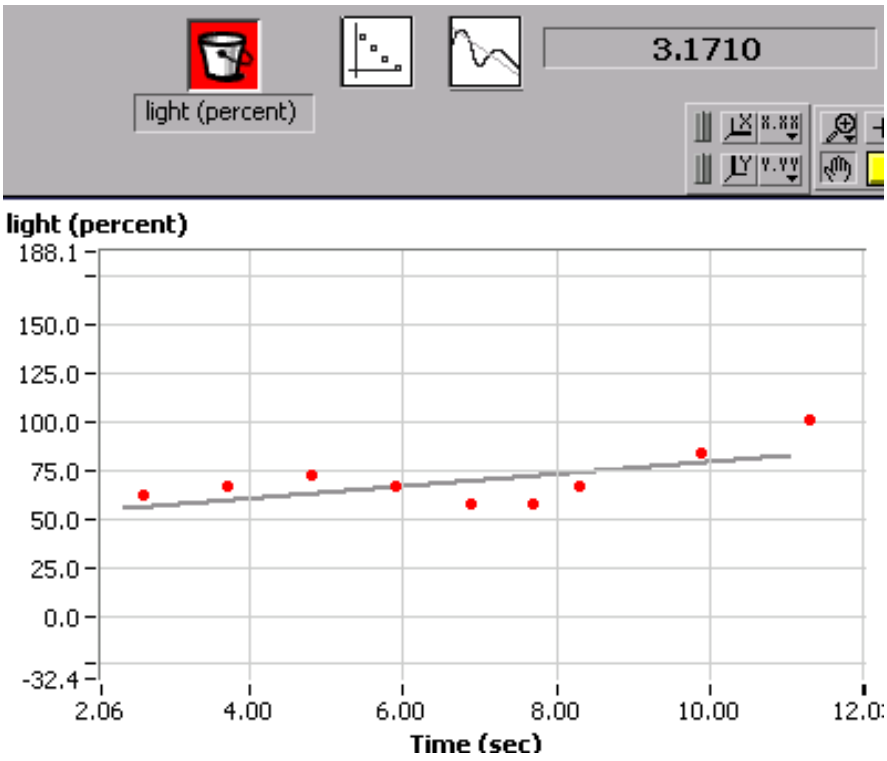
The up load screen will look like this once the information is up loaded with the correct program slot being used of course.

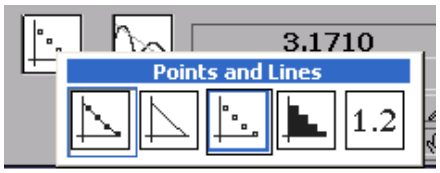


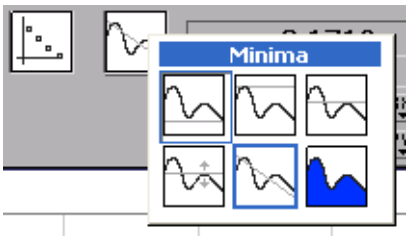

High lighting the page name can change the name of this page and typing in whatever name you want to call it once you have this done it is also a good idea to add the page to the journal by selecting the + button at the bottom of the main menu. To delete a page from the report select – and to view the journal click on the extreme right icon.

Once you have added the up loaded page to your journal you may need to view and compare readings this is achieved by selecting the buckets/bins. You have the option of viewing the data graphed or logged, comparing it to different findings or measuring your data. In the sample supplied on CD you can see I

have shown the (1) graph (2) lowest % light (3) highest % light (4) standard deviation (5) slope of best-fit line. The diagram below shows the slope of line with best fit. The pages can be added accordingly to change the slop graph for example to the lowest % light graph the two graph icons are changed by selecting and clicking on the desired format.

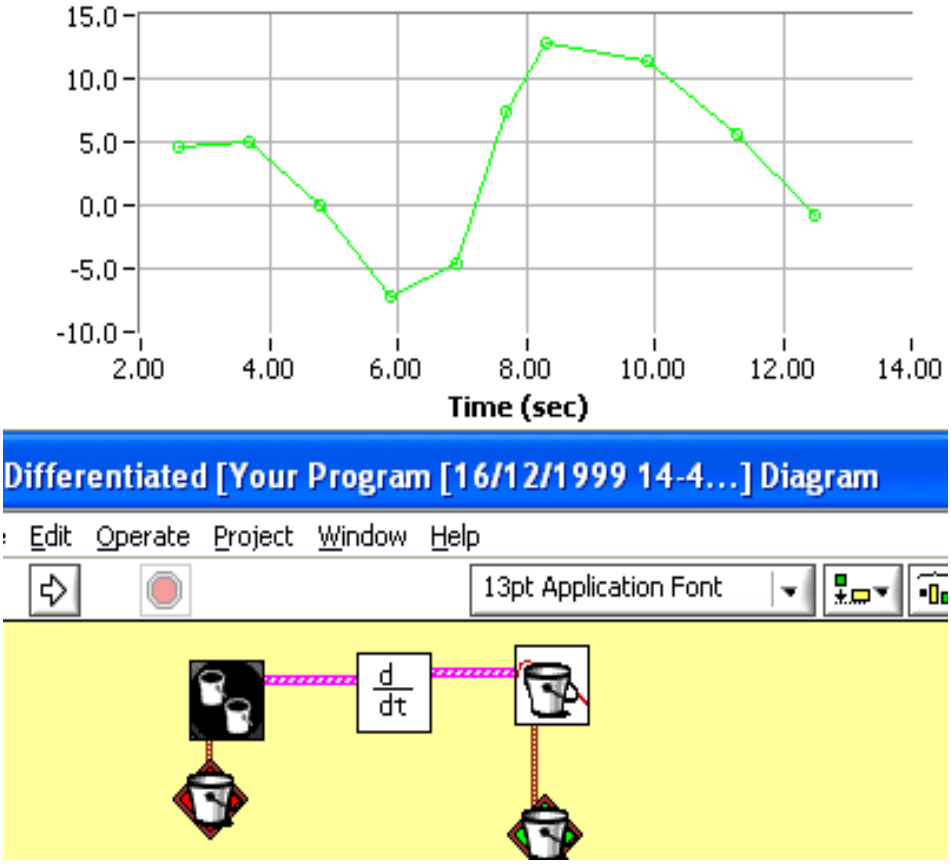



Above are the graph icons in the Measure, view and compare area.

In the next area is the compute section. There are 5 compute tools each one getting more difficult as they ascend. The compute area allows you to work mata-matically with the data that you have collected. In the example on the CD the graph is differentiated and this is produced by programming the Light values (Red Bin) to d/dt = output (green bin). A lot more complex maths can be carried out using the G code in Compute Tools level 5. This can be seen in the program screen below.




You should access the journal area as often as possible when configuring your data, as it is a clear means of debugging any problems that you may have made but have not noticed. For example some pages of the report might have noting on them.

The first report page shows the contents page of the report. You are shown three options/icons you can print, publish to the web or view a power point display of the report. When you select the power point slide show (computer screen) a description of the project page will come up. To enter text into this and a picture it is possible to go back to the main menu and click on the centre icon that displays an open book. In this area you can describe the main areas of your research and your rationale and findings or even include a picture.

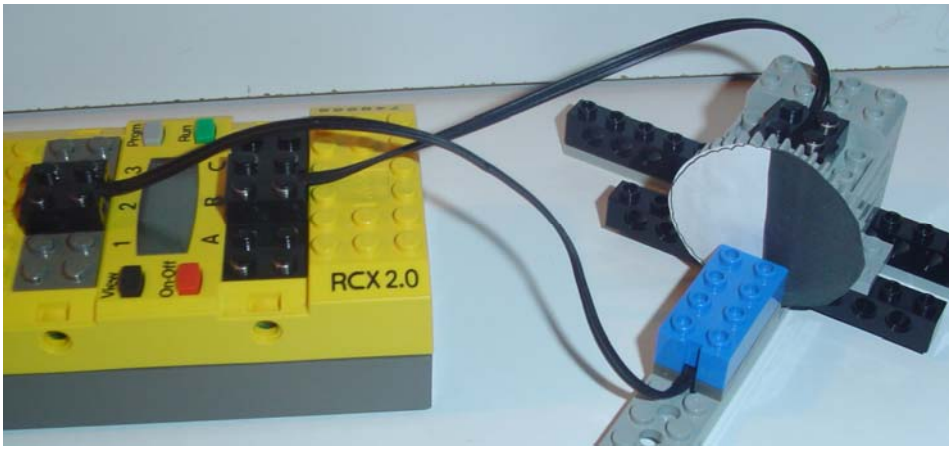
## Module 24: Investigator design brief

Using the RCX, one light sensor and a motor calculate the speed of the motor at power level 5 and the motor at power level 1.

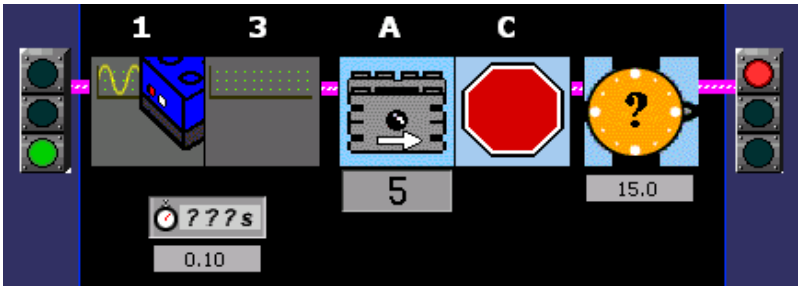
To interrupt this problem we should firstly break it down. Consider the first problem calculate the speed of the motor at speed 5. The light sensor needs to be used to find the rotation of the motor as that is going to be the speed we are looking for (Rotational speed).

How can we use the light sensor to count how many times the gear turns around? What sort of program will we need to program the sensor to count how many revolutions occur.

The answer to this solution is quite simple affix a large (40 tooth) gear on to the motor. Create a disk the same size out of paper with one half white and the other half black. Using blue tack stick this disk on to the front of the gear and place the light sensor in front of the gear. Therefore when the gear rotates the sensor can note the change in light percentage. When the light sensor values are plotted against the time in seconds it will be possible to count the speed in seconds.

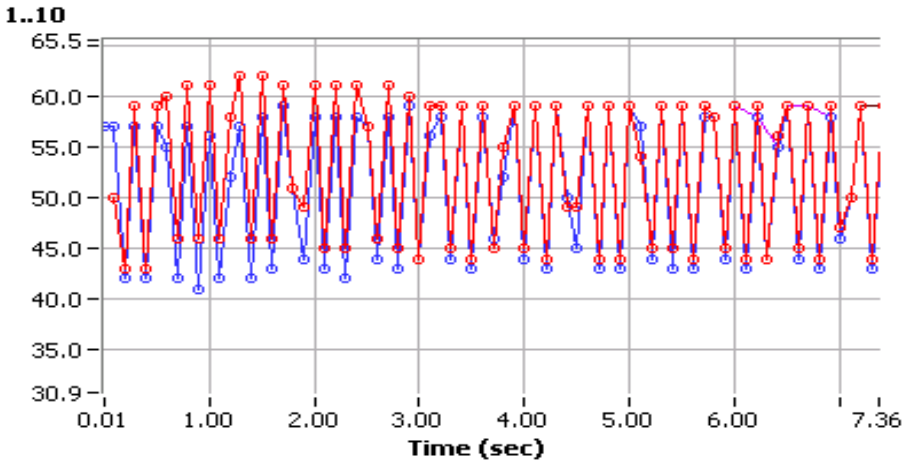


The program is designed in program level 2 and is shown below.

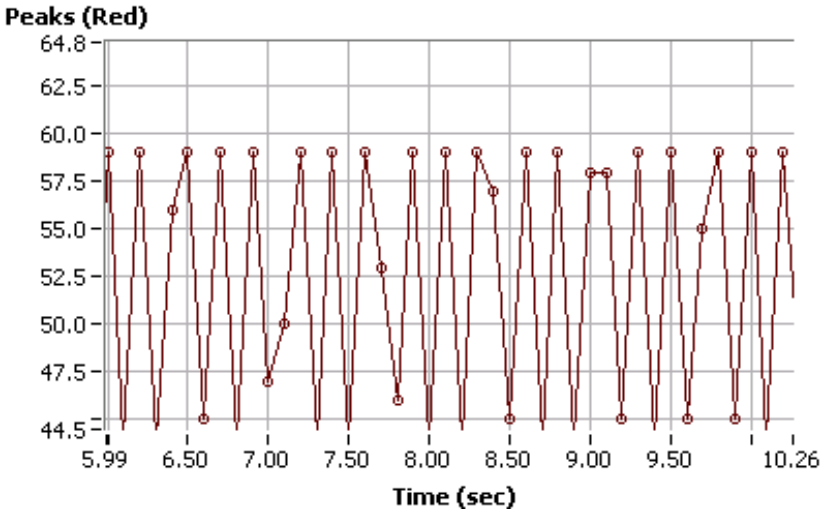



Notice the time the light sensor is set to record the light percentage values. This tenth of a second means that every second 10 readings will be taken for 15 seconds giving a data log of 150 points. If the time was set to anything higher it may miss revolutions of the gear and for really accurate readings the time could be lowered. The motor is set to power level 5 also.

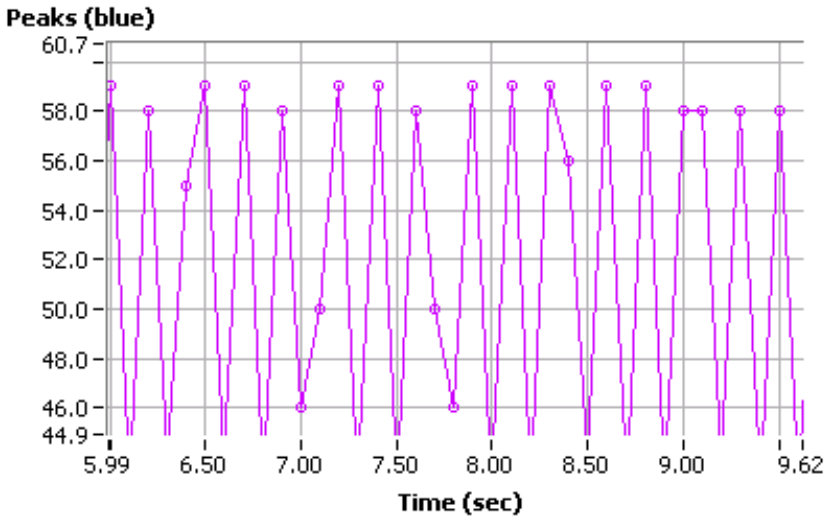
Once the material is up loaded to the computer you can view your findings in the View and Compare area. In my report I compared the readings that I took to see would there be a difference and there was. This change did not affect the motor speed. Only the light percentage changes and this was due to me turning on the light in the room for the data collecting. Can you see which graph had the light on during the data collecting (red) (blue)?



When the data is compared it will be necessary to zoom in to find out how many peaks are in a second. To get a reliable result you should take 2 seconds and divide the sum total of peaks by 2. This result will give you the rotations per second. Both data types' blue and red samples should be taken to find correlation.




Between 6 and 8 seconds the peak value is 9. This equals 4.5 revs per second.




Between 6 and 8 seconds the peak value is 9. This also equals 4.5 revs per second. To find the revs per minute is 60seconds per min X 4.5 revs = 270 revs per min.

Upon further investigation of the graph form the 0-2 seconds slot, why do we not take these slots as our calculation areas?

You should now find out what speed the motors rotates per min at the power level 1.

## Investigator design briefs

1. Demonstrate Simple Harmonic Motion (SHM) using the RCX and a light sensor. An elastic band or spring may be used. A lamp should be situated at the end of the table so you can tabulate light intensity against time and the curve should represent a simple harmonic motion curve.

2. Calculate the acceleration of gravity by using the RCX and a light sensor. The pendulum method should be used and this can be further researched in most physics leaving cert books.

3. Design a device with two touch sensors that will test a person's music ability. When a note is played the person can either click sensor 1 or sensor 2? When a note form the RCX is higher than a preset note the sensor 1 should be touched and when a lower note is played sensor 2 should be pressed. The RCX should record these results and graph the correct and wrong responses.

## Related Web sites

1. http://www.sip.ie/sip069/Documents/engineeringpage.html

2. http://www.southwest.com.au/~jfuller/lego/lego.htm

3. http://hces.hunter.cuny.edu/h/robotics/

4. http://carol.wins.uva.nl/~leo/lego.html

5. http://www.galileo.org/robotics/design.html

6. http://www.play-hookey.com/digital/electronics/

7. http://www.ceeo.tufts.edu/graphics/robolab/Intro.htm

8. http://www.tvcc.cc/staff/fuller/cs281/Lecture.htm

9. http://www.ceeo.tufts.edu/

10. http://www.teachers.ash.org.au/dbrown/robolab/rcx.htm

11. http://www.visi.com/~dc/

12. http://mindstorms.lego.com/eng/default.asp

13. http://www.tep.org.uk/index2.html

14. http://homepage.eircom.net/~steppermotor/