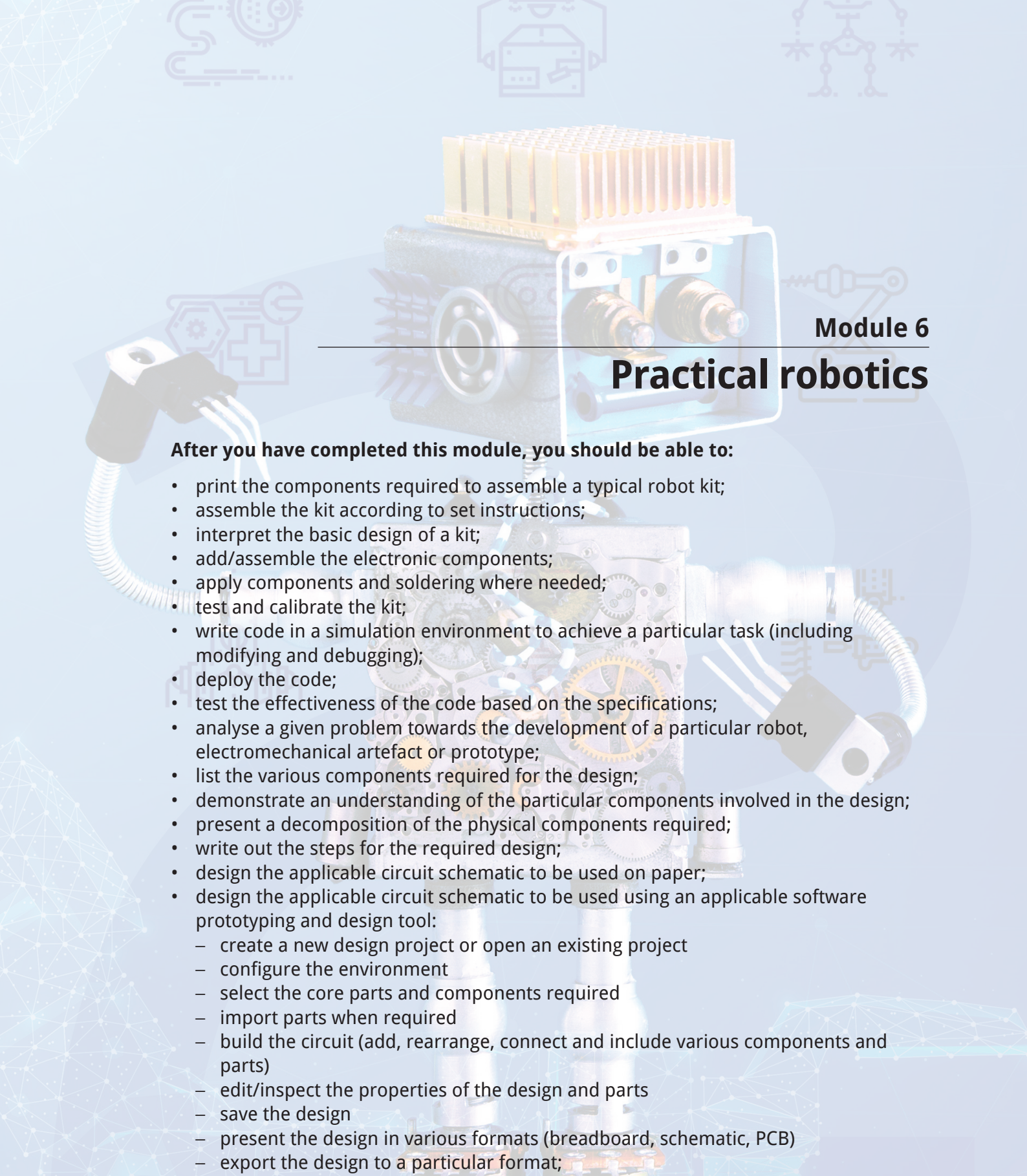


Module 6

Practical robotics

After you have completed this module, you should be able to:

- print the components required to assemble a typical robot kit;
- assemble the kit according to set instructions;
- interpret the basic design of a kit;
- add/assemble the electronic components;
- apply components and soldering where needed;
- test and calibrate the kit;
- write code in a simulation environment to achieve a particular task (including modifying and debugging);
- deploy the code;
- test the effectiveness of the code based on the specifications;
- analyse a given problem towards the development of a particular robot, electromechanical artefact or prototype;
- list the various components required for the design;
- demonstrate an understanding of the particular components involved in the design;
- present a decomposition of the physical components required;
- write out the steps for the required design;
- design the applicable circuit schematic to be used on paper;
- design the applicable circuit schematic to be used using an applicable software prototyping and design tool:
  - create a new design project or open an existing project
  - configure the environment
  - select the core parts and components required
  - import parts when required
  - build the circuit (add, rearrange, connect and include various components and parts)
  - edit/inspect the properties of the design and parts
  - save the design
  - present the design in various formats (breadboard, schematic, PCB)
  - export the design to a particular format;
- build or prototype of the applicable circuit with applicable components;
- develop, code and debug the applicable source code for the operations of the artefact;
- discuss the composition of the code;
- move the applicable code to the controller;
- finish and present the design and artefact;
- test the artefact against a set of criteria; and
- create a user manual and a description of the operation of the artefact.

Introduction

As technology improves, we will depend more and more on robots and artificial intelligence to make our lives easier, better and/or safer. There is no better time to learn robotics, whether to satisfy your inquisitiveness or as your future career. Although many people think robotics is highly complex and very difficult for inexperienced people to enjoy, recent advances have made robotics more accessible than ever. In this module, you will be introduced to robot kits designed to help you to develop and build your own fun, functional or even silly robots. Skills such as assembling electronic components and interpreting circuit designs will be revised. You will also be equipped with the necessary programming skills to test, modify and debug the artefact or robot you have built.

6.1 Practical robotic projects

A *robot kit* is a special construction kit for building a robot. Kits range from basic beginner kits all the way through to advanced robotics that offer Raspberry Pi control and clever sensors to measure a variety of conditions.

Examples of kits include the following:

- Turtle: a two-wheel drive Arduino robotics kit for beginners
- Alphabot2, Raspberry Pi robot kit
- Alphabot2, robot kit for Arduino with Uno Plus
- Alphabot, Raspberry Pi robot kit
- Picogo mobile Raspberry Pi robot kit.

In this section you will learn how to interpret the basic designs of a kit and assemble a typical robot kit according to the designers' instructions. The next step involves an exciting journey into electronics and programming. Get ready to build your own robot! All electronic components can be purchased online or in specialist shops.

6.1.1 Components required to assemble a typical robot kit

As mentioned above, a typical robot kit contains what you need to construct a robot. This varies, depending on the user's level of experience and of course how much he/she is prepared to spend. Basic kits may include a motor, small programmable microcontroller, sensors and more. Additional parts are available separately if you want to expand your robot's capabilities.

The list of components below is by no means complete and you may find many variations.

Components required are the following:

- Raspberry Pi 3 Model B+ (usually not included in the kit – read the specifications)
- 8GB+ microSD card
- HDMI cable

- USB keyboard/mouse
- 5-V micro-USB power adapter
- 400-point breadboard
- An LED with appropriate resistor
- Male-female/Female-female/Male-male jumper wires
- Momentary pushbutton switch
- Chassis for your robot (can build one using LEGO)
- Two-brushed 5-V to 9-V motors with tyres
- Six AA-battery holder
- Six AA-batteries (recommend rechargeable)
- LM2596 buck-converter module
- L293D motor-driver chip

In the subsections that follow you will be guided through the assembly process: how to connect the electrical components to get your robot to follow instruction. The Raspberry Pi will be used for all the projects.

6.1.2 Assemble the kit according to set instructions

Every kit will have a specific set of instructions that should be followed exactly if you want your prototype to do what you have designed it to do.

Before getting started, you will need a Raspberry Pi. There are numerous types of Raspberry Pi models available. The Raspberry Pi 3 model B+ and the Raspberry Pi Zero are recommended for this course as they are the best options for development and offer more full-sized connectors. It does not really matter which version of Raspberry Pi you have since the models are all compatible and can be used to create the specified robot/artefact. You may need adapters for functions such as wireless Internet connection.

The table below sets out the specifications different Raspberry models.

MODEL	RAM	PROCESSOR	PORT	CONNECTIVITY
Raspberry Pi 3 Model B+	1 GB	64-bit quadcore 1,4 GHz	HDMI, 4 × USB 2.0	Wi-Fi, Bluetooth, Ethernet
Raspberry Pi Zero	512 MB	32-bit single-core 1 GHz	Mini HDMI, micro USB (data), micro USB power	None
Raspberry Pi Zero W	512 MB	32-bit single-core 1GHz	Mini HDMI, micro USB (data), micro USB power	Wi-Fi, Bluetooth

Table 6.1: Comparison of Raspberry Pi specifications

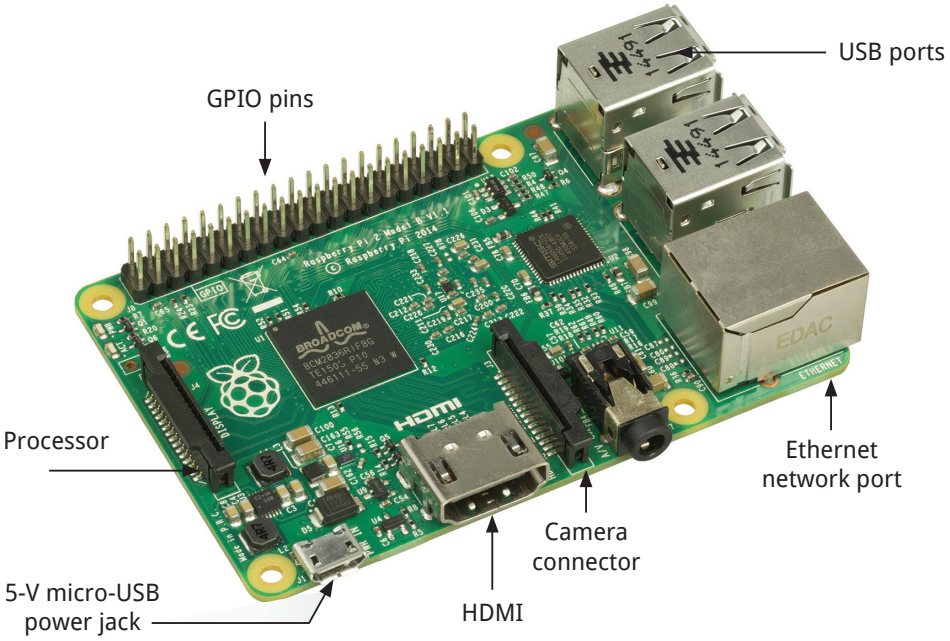


Figure 6.1: Raspberry Pi and its components

Let us revise the different components and what they can do – refer to Module 4 for more details regarding the various components. When a Raspberry Pi is unpacked, you will find a bare-looking board with all sorts of components sticking out of it. It should look more or less like the figure above.

- USB ports
  - Number may vary
  - Used to plug in USB keyboard, mouse, USB stick and other devices
- Ethernet network port
  - Used for wired Internet connection
- HDMI (high-definition multimedia interface) port
  - Used to connect the Pi to a screen like a TV or a computer monitor
- Micro-USB power jack
  - To connect the Raspberry Pi to 5 V of power
  - There is no power button; the Pi will stay on for as long as it is connected to the power cable
- Processor
  - This is the brain of the Raspberry Pi
  - The processor coupled with 1 GB of RAM gives the Raspberry Pi power roughly equivalent to some smartphone
- Camera connector
  - Input for the Raspberry Pi connector
  - Peripheral hardware can be used to give your robot the ability to see
- GPIO pins
  - The 40 metals pins called *general purpose input output* or GPIO (revise Module 4)
  - Can be programmed to control a variety of electronic components such as LEDs, sensors and motors
  - The gateway to the world of physical computing – you will use them to wire the electronic components of your robot (motors, breadboard, sensors, etc.)

NOTE

The microSD *slot is not shown in Figure 6.1. The Raspberry Pi has no onboard storage. The software used to run it is known as the* operating system OS *and all your files are stored on a microSD card (revise the terms in Module 4).*

[NB: Heading – assemble the kit according to set instructions yet NO instructions given? Shouldn't the info re. the RPi. GPIO module be given here? Do they even know what a 'module' is?]

Assembling the kit

Make sure you have downloaded and installed the appropriate program and run it on your computer – refer to the section 6.1.7.

Start by looking at the various components included in the kit. For example, if you are using swivel wheels, the instructions will list the parts needed and the steps that should be taken to assemble the wheels.

Gather all the tools you will need, e.g. soldering iron and solder, jumper leads, batteries, microcontroller. Follow the steps for attaching the motor and wires and connecting the wires to the motor controller terminals. Next you would have to mount specific parts as directed, using the fasteners provided in the kit and the method described.

Finish off by connecting the motor controller board with the Raspberry Pi according to the type of motor controller you are using. Connect the specified pins with wires to the Raspberry Pi.

To give the robot direction, you would use *mu* in the Raspberry Pi programming menu, as set out in section 6.1.7.

Finally, assemble or create the robot's body. Here you do not have to follow strict guidelines, as long as you make sure all the components fit perfectly. If you are using two motors, you would attach the left and right motors and one end of the body (back) on either side, corresponding to the left and right of the robot. The controller and the Raspberry Pi will sit in front of them at the other end. If you are using wheels, make sure the spinning components of the motors stick out on either side so that you can attach the axels and wheels to them.

NOTE

Mounting the motor(s) and other electronic components is discussed in section 6.1.4 and section 6.1.5.

6.1.3 Interpret the basic designs of a kit

[I'm sorry, but I don't understand the difference between 6.1.2 Assemble the kit according to instructions and this section. The info that follows seems to fall under 6.1.2? PLEASE CHECK!]

To get started with physical computing on Raspberry Pi, you will be using a programming language called *Python*. The RPi.GPIO package provides a Python **module** used to control the GPIO interface on the Raspberry Pi. It is the driving force behind Python and will interpret how the GPIO pins are used to interface with electronic components such as LEDs and PIRs, sensors, wheels and many other devices.

Remember: pin numbering declaration ?What is this and where is this given?
{NO mention has been made of the RPi.GPIO module? Doesn't this belong under previous heading?]

After you have included the RPi.GPIO module, the next step is to determine which of the two pin numbering schemes you want to use:

- *GPIO BOARD* – This type of pin numbering refers to the number of the pin in the plug, that is the numbers printed on the board. The pin numbers follow the pin numbers on header p1.
- *GPIO BCM* – This refers to the pins that are directly connected to the system on a chip of the Raspberry Pi. These pin numbers follow the lower-level numbering system defined by the Raspberry Pi to connect sensors and components directly to the brain of the Pi.

VOCABULARY

Module – *any of several distinct but interrelated units from which a program may be built up/into which a complex activity may be analysed*

To specify in your code which number system is being used, use the `GPIO.setmode( )` function. For example:

`GPIO.setmode(GPIO.BOARD)` – This will activate the broadcom chip-specific pin numbers. Both the import and set mode lines of code are required, if you want to use Python.

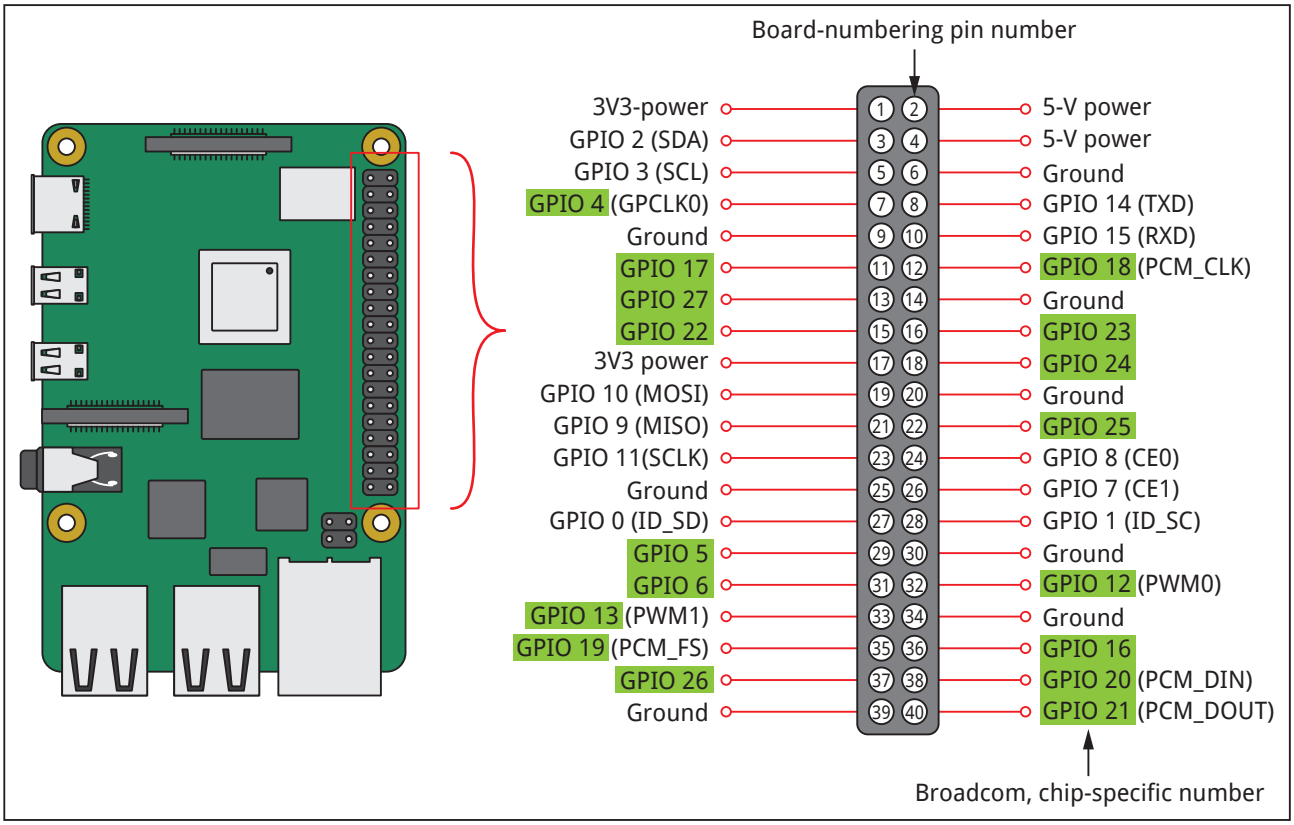


Figure 6.2: GPIO pins and numbering systems

REMEMBER

The pin number will change if you are using the board numbering system (instead of 18 it would be 12).

Set a pin mode

If you are using Arduino, you will probably be familiar with the fact that you have to declare a *pin mode* before you can use it as either an input or output pin. To set a pin mode, use the `GPIO.setup(PORT_OR_pin)` function and enter `GPIO.IN` or `GPIO.OUT`.

If you want to set pin 18, you would write:

```
Gpio.setup(18, gpio.out)
```

[MUST BE EXPLAINED; CHECK USE OF CAPITAL LETTERS]

Output

Digital output

To write a pin high (1) or low (0), use the `GPIO.OUT(specify port_or_pin), [GPIO.LOW, GPIO.HIGH])`.

For example, if you want to set pin 18 high, write:

```
GPIO.output(18, GPIO.HIGH)
```

Writing a pin to *GPIO.HIGH* will drive it to 3,3 V, and *GPIO.LOW* will set it to 0 V. Alternatively, use 1 for high (true) and 0 for low (false).

Configure the environment

NOOBS is an OS installation program created by Raspberry Pi. It is minimal disk image that can just be copied onto a newly formatted SD card.

Prepare the SD card

Before installing the Raspbian on the microSD card, format the SD card to delete anything that might be installed on it. Even if the microSD card is brand new, this is strongly recommended.

Step 1: Insert the SD card into the computer. Some computers have SD-card or microSD-card ports. If the computer you are using does not have a port, you will need to use a USB SD-card adapter. This device will enable you to plug your card into the USB slot on your PC.

Please draw both.

Figure 6.3: Inserting the SD card

Figure 6.4: USB SD-card adapter

Step 2: Once the SD card has been inserted, use the file explorer to navigate and locate the microSD card. Make a note of the drive name used for the card – that is the letter your computer assigns to the SD card when it plugged, such as `D:` or `E:`.

An 8 GB or one with greater capacity is recommended with Raspberry Pi OS. If you are using a light version of Raspberry Pi OS, you can use a 4-GB card. Check with the supplier of the OS to find out the recommended capacity.

Step 3: Ensure that the SD card is completed wiped and formatted appropriately. There is software that can assist you with the formatting. Download and install the SD memory card formatter and find the formatter for the OS you are using. Accepts terms and conditions.

eLINK

To download and install an SD card, visit the following link:
futman.pub/SDinstallation

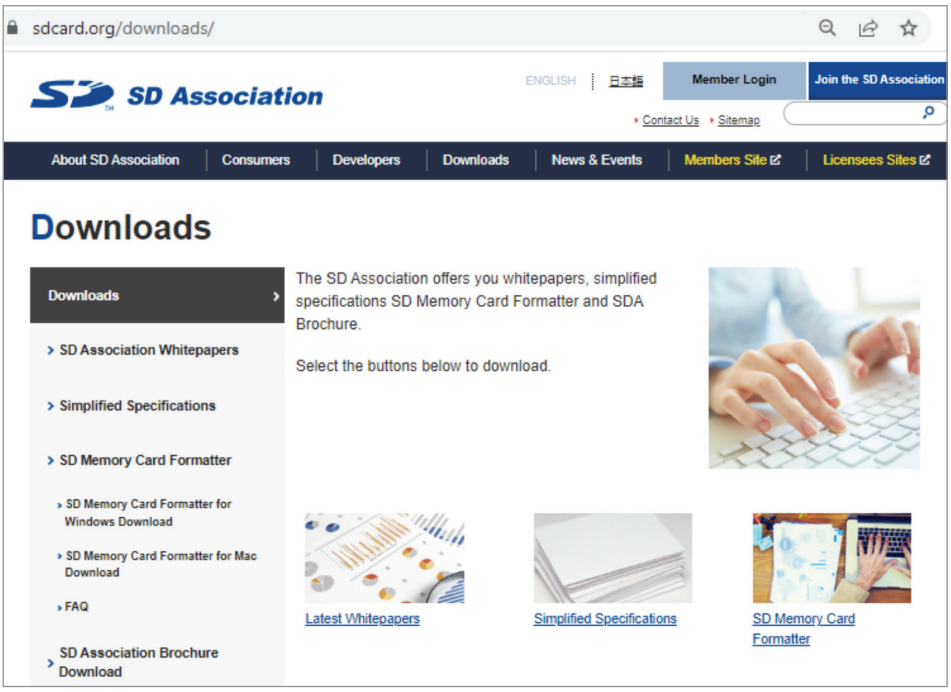


Figure 6.5: Link to SD card

Step 4: Once the installation is completed, locate and run the microSD card formatter. The process from here on is easy:

- Select the card from the dropdown menu (remember the drive name as mentioned earlier).
- Keep the option `Quick Format` selected, then click `Format`. Watch the progress bar as your card is successfully formatted.

Installing Raspberry Pi imager (OS)

The Pi imager, also known as *Raspbian (OS)*, is a small image that can be downloaded from the Internet for free.

eLINK

To get a copy of Pi imager, visit the following link:
futman.pub/RaspberryPi_OS

6.1.4 Assembling the electronic components

In this section you will learn how to identify the critical GPIO features and connect a variety of components, including *IoT* or *internet of things* devices, to the Raspberry Pi. What is more, you will interact with the components using Python programming language.

GPIO pins

GPIO pins are discussed in the previous sections (revise section 4.1.9 of Module 4). When the pins are connected to electronic circuits, the Raspberry can control LEDs (turning them on or off), run motors, detect whether a switch has been pressed or temperature and light intensity has changed, among other things. We refer to this as *physical computing*.

If the pins are labelled, it can help you identify what each pin is used for. Make sure your pin label is placed with the keyring hole facing the USB ports, pointing outwards. If you do not have a pin label, refer to section 4.1.10 in Module 4, where the pins and their functions are discussed.

REMEMBER

The following pins serve a very specific purpose:

3V3	*3,3 volts*	*Anything connected to these pins receive 3,3 V of power*
5V	*5 volts*	*Anything connected to these pins receive 5 V of power*
Ground	*0 volts*	*Used to complete a circuit*

Selection of hardware components that can be connected to the Raspberry Pi using the GPIO pins

You will need to assemble a variety of electronic components to complete the circuit and make your robot react. These include the following:

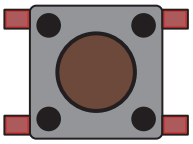
- Through-hole passive components
  - Resistors
  - Capacitors
  - Jumpers
  - Inductors etc. and the
- Through-hole active components
  - Diodes
  - Transistors
  - ICs.

NOTE

Refer to section 4.2.10 to revise active and passive electronic components. The most important components are listed below.

You will need the following components:

- Solderless breadboard – enables you to connect electronic components to produce a functional circuit (revise section 4.2.5 in Module 4)
- Pushbutton switch – an on/off electronic switch that closes the circuit when pressure is applied (the button is pressed); as soon as the switch is released, the circuit is broken (open) and no current flows through it

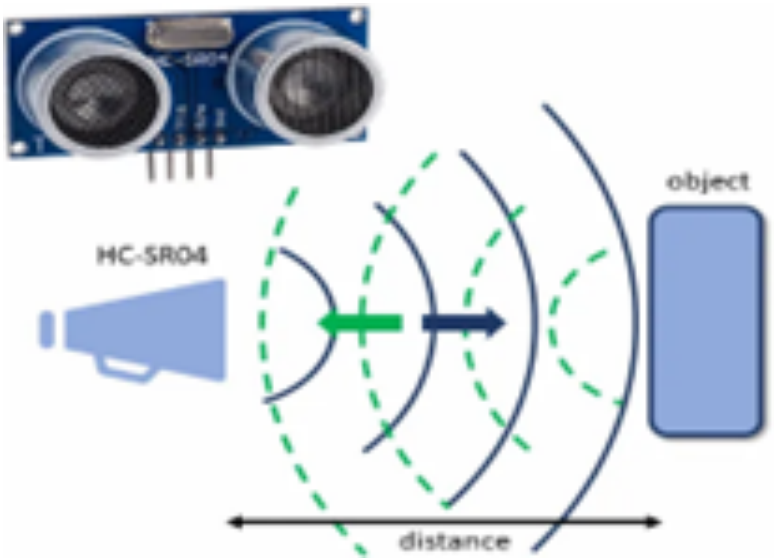


- LEDs – emit light when current flows through it

NOTE

Distance sensors are useful for many projects, especially outdoor prototypes where a distance measurement is necessary or advantageous. These sensors can measure distances of up to 4 to 5 metres and are surprisingly accurate.

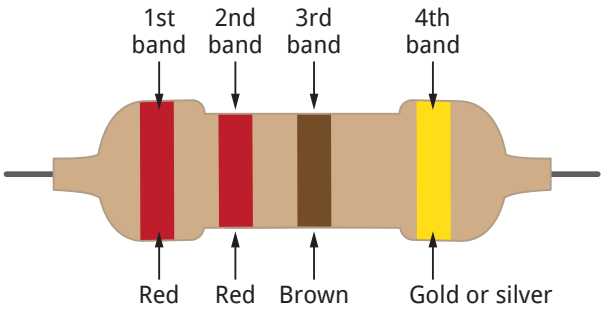
- Ultrasonic distance sensor – an electronic device that measures the distance of target objects by emitting ultrasonic sound waves and converting the reflected sound into an electrical signal



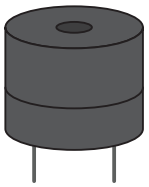
REMEMBER

The 220-Ω resistor is one of the most common resistors in electronics. The value and tolerance of colour-coded resistors can quickly be determined just by looking at the colour bands on the body of the resistor.

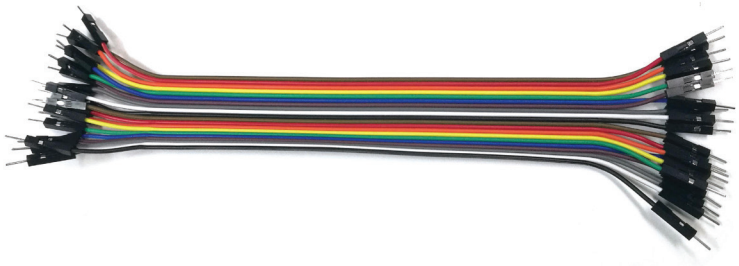
- 3 × 220-Ω resistors (red, red, brown, gold) – to resist the flow of electrical current



- Buzzer – audio signal device used as alarm devices, timers or to indicate other changes in conditions in the circuit



- Jumper wires – make connections to create electronic circuits



The physical setting up of the Raspberry Pi is your next step. You will require an area that has space and access to a monitor whether a **[MISSING TEXT]**

6.1.5 Apply components and soldering where needed

NOTE 1: Syllabus point – assemble the electronic components. Nothing here re. assembly. Components are merely listed – repeating info in Module 4?

NOTE 2: Are they using PCBs or breadboard in the assembly? Doesn't the soldering refer to the components that may need soldering?

NOTE 3: I have inserted the soldering info below – EDCR Module 7 – because students may not be familiar/may need to revise this?]

A PCB is the heart of electronic equipment. Without it, electronic equipment, cell phones and computers would not exist. PCBs can have one layer or many layers, depending on the complexity of the project. Single-sided circuit boards use through-hole components whereas multi-layered boards are likely to use surface-mounted components. Multi-layered PCBs using surface-mounted devices can be populated much more compactly.

Soldering can be defined as a low-temperature joining method in which the **solder**, which is the joining material, has a much lower melting point than the surfaces to be joined. The process involves using solder to join metallic surfaces without melting them. The result is a solid conductive joint. Tin-lead solder was once used by everybody, but lead-free solder is now the preferred type of solder, because it is less toxic and more environmentally friendly.

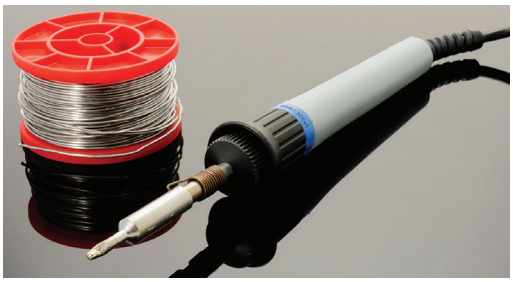


Figure 6.6: Solder and soldering iron

When you want to remove a faulty component from a PCB, you will need a soldering iron and de-soldering gun or solder sucker.

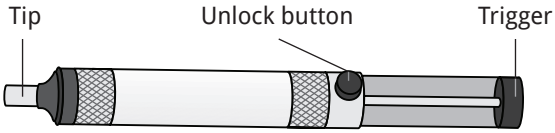


Figure 6.7: Solder sucker

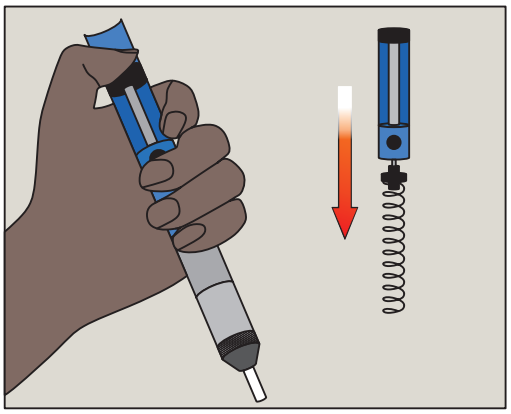
VOCABULARY

Solder – *an alloy (usually with a lead, brass, tin or silver base) with a low melting point; used for joining metals*

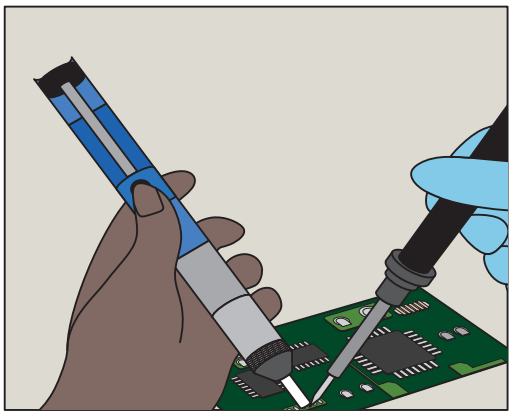
De-soldering

De-soldering is done as follows:

- Heat the soldered area using the soldering iron.
- Once the solder has melted, press down the trigger of the solder sucker.



- Place the tip of the solder sucker against the solder you want to remove.



- Press the unlock button to suck up the solder. You can remove the freed component.
- Repeat these steps to remove any excess solder.

DID YOU KNOW

Solder has a melting point of 90 °C to 450 °C.

The following very important safety precautions must be taken when you solder or de-solder components:

- The soldering iron gets very hot (± 400 °C). It must be handled with care.
- Never leave a soldering iron lying on the workbench. Always put it back into its stand when it is not in use to prevent it from causing fires.
- Lots of harmful fumes are generated during the soldering process, so work in a well- ventilated room
- Hold wires/Components to be soldered with tweezers or clamps to prevent burns.
- Turn the soldering iron off or unplug it when it is not in use.
- Always use lead-free solder.
- When you are soldering bits of material may fall. Never try to catch hot material with your bare hands.
- Always wear safety glasses when you solder.
- Take care not to let the nozzle of the solder sucker touch the hot tip of the soldering iron, as this will melt the plastic nozzle.
- When de-soldering, take care not to apply too much heat to the joint, as this may damage the sometimes-thin copper tracks.
- After handling solder make sure you wash your hands thoroughly.

Tools

The following basic tools are required to successfully assemble a PCB:

- Small wire cutter
- Small pliers
- Soldering iron – 15 to 30 watt
- Solder
- Anti-static mat and wrist strap
- Solder sucker
- Damp sponge
- Multimeter (volt-/ohmmeter)

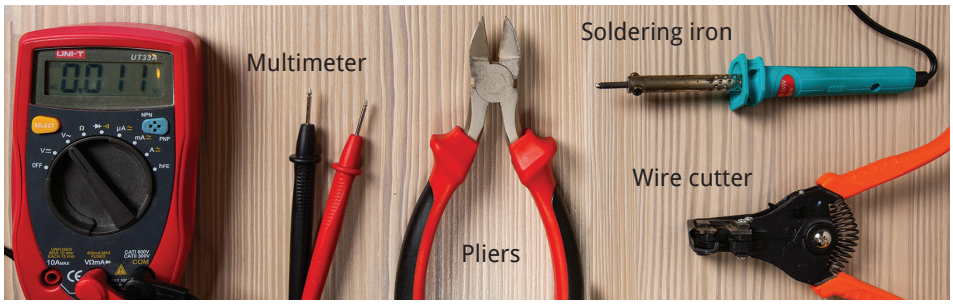


Figure 6.8: Soldering tools

DID YOU KNOW

Anti-static wrist straps and mats are important safety measures. They prevent the user from accumulating static charge that could damage the electronic assembly or component being used. The wrist straps help to disperse static electricity generated by the user safely to the ground.

There are many types of soldering devices to choose from. Thermostatically controlled irons have several ways to regulate the soldering temperature. When you choose a soldering iron, make sure the tips can be replaced.

Type of solder

There are two categories of solder for electronics work: lead-tin **alloy** and lead-free. Lead-free solder is an alloy of tin, silver and copper. The melting temperature of lead-tin solder is around 183 °C and that of lead-free solder about 217 °C, so either of these is suitable for electronic work. Solder containing lead is cheaper and easier to use. Since little solder is used, it does not present pollution or health issues.

VOCABULARY

Alloy – *metal made by combining two or more metallic elements*

Appropriate soldering technique

The correct soldering technique involves heating the surface being soldered beyond the melting point of the solder used so that the solder flows freely onto the surface.

Steps to follow:

1. Clean the tip of the soldering iron by filing it until it is smooth and clean.
2. Heat the soldering iron.
3. Apply the solder directly onto the tip of the soldering iron.
4. Apply the soldering iron to the joint and allow the solder to run into the joint.

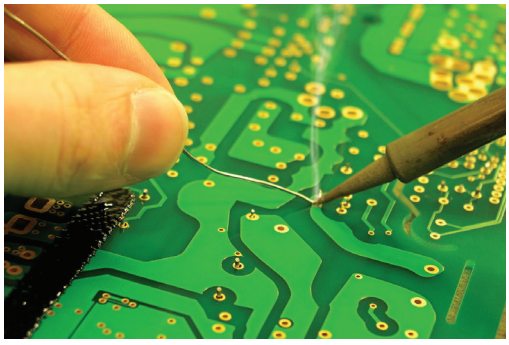


Figure 6.9: Soldering a PCB

REMEMBER

Poor joints are usually the result of the following:

- *Not heating the joint for long enough*
- *Not cleaning the soldering iron and/or* tinning *the tip*
- *Poor soldering iron contact*
- *Moving the components before the joint has cooled.*

VOCABULARY

Tinning – *covering the tip with a thin layer of solder*

6.1.6 Test and calibrate the kit

[Info missing]

6.1.7 Write code in a simulation environment to achieve a particular task (including modifying and debugging)

[NOTE: No example of codes/ how it is written; also no info on actual modifications of codes or debugging.]

In this section, we will investigate Python as an interpreted, high-level, general-purpose programming language. Python has been around since 1991. Presently it is the most popular and fastest-growing programming language. Python interpreter is the recommended language on the Raspberry Pi platform. There is another programming language for beginners that can be used with the Raspberry Pi called Scratch. It is simple to use. You just drag the code you have selected and drop the block in the coding area to build your program.

eLINK

To learn more about Scratch, visit the following link:
futman.pub/Scratch

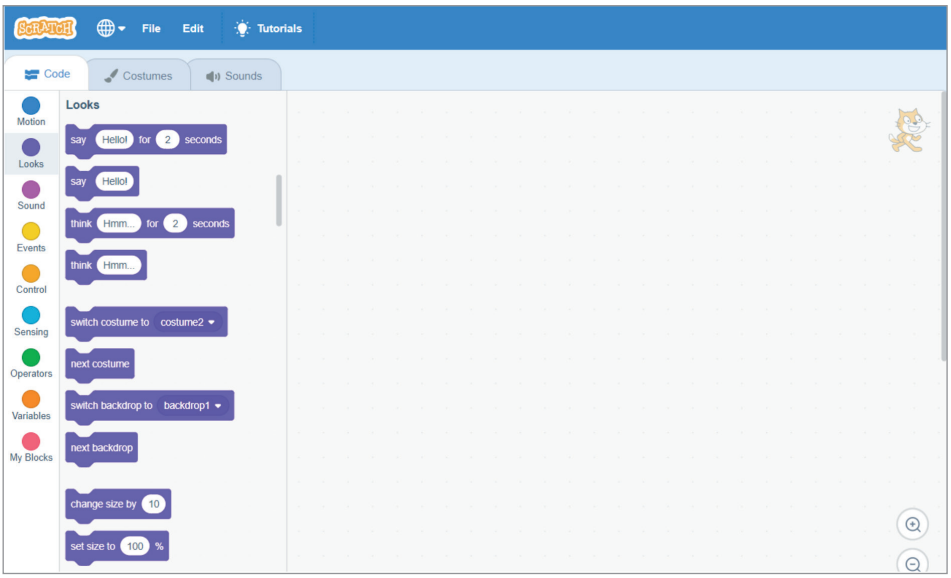


Figure 6.10: Online Scratch for beginners

Python

Python features enable the interpreter to type and try commands interactively without the need to create a program. The interpreter does not need to explicitly compile programs. They are compiled at run time in an intermediate bytecode which is executed by a virtual machine. The model code in this section is written for Python and should work on any Raspberry Pi model.

Python is a module that controls the GPIO interface on a Raspberry Pi. The RPi.GPIO is installed by default on recent versions of Raspbian Linux. To use a Python module, it must first be imported. Import RPi.GPIO as GPIO.

DID YOU KNOW

Guido Van Rossum is the author of the general-purpose programming language Python*, which was released in 1991. It is used in web development, data science, creating software prototypes and in many other areas.*

Getting started with mu

This section focuses on how to use the *mu* code editor. *Mu* is a simple, easy-to-use Python editor and integrated development environment (IDE) for beginners. It is designed to be as user friendly and helpful as possible.

Go to the Programs menu and click on *mu*, as illustrated below. This will open the *Python editor for beginner programmers*.

RE SCREENSHOTS
Some of these screenshots are very badly taken/ quality/blurry. Please provide new clearer ones. CHECK THROUGHOUT

Figure 6.11: Opening mu

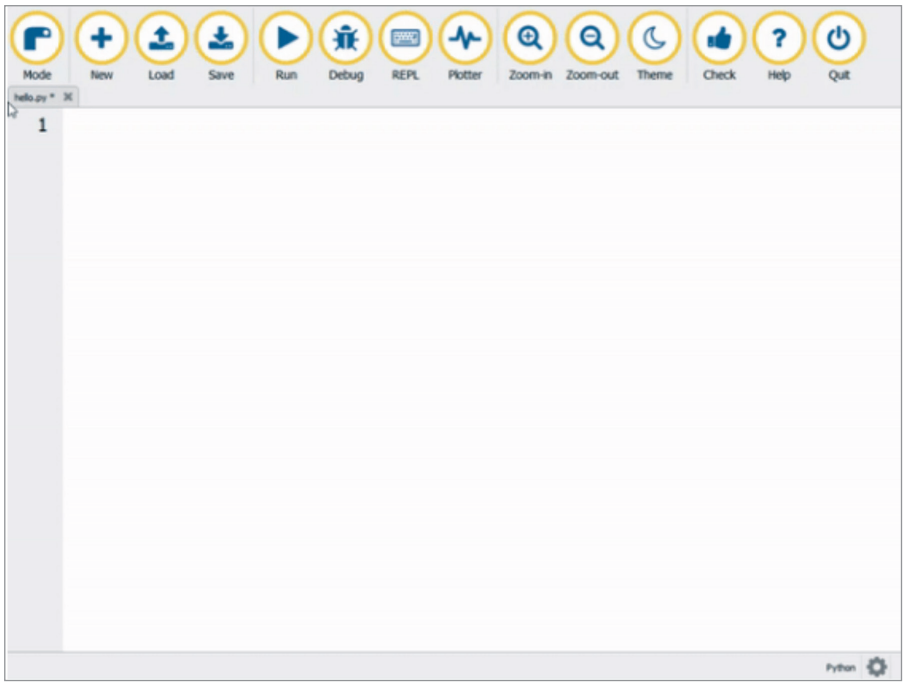


Figure 6.12: Mu graphical user interface environnent

Mu mode

Mu can be used in various modes, e.g. a mode that makes it easy for junior programmers to use Mu by only presenting the options most relevant to what he/she is planning to do. When you open Mu for the first time, you will be presented with the mode screen, as illustrated below. Select the mode of your choice.

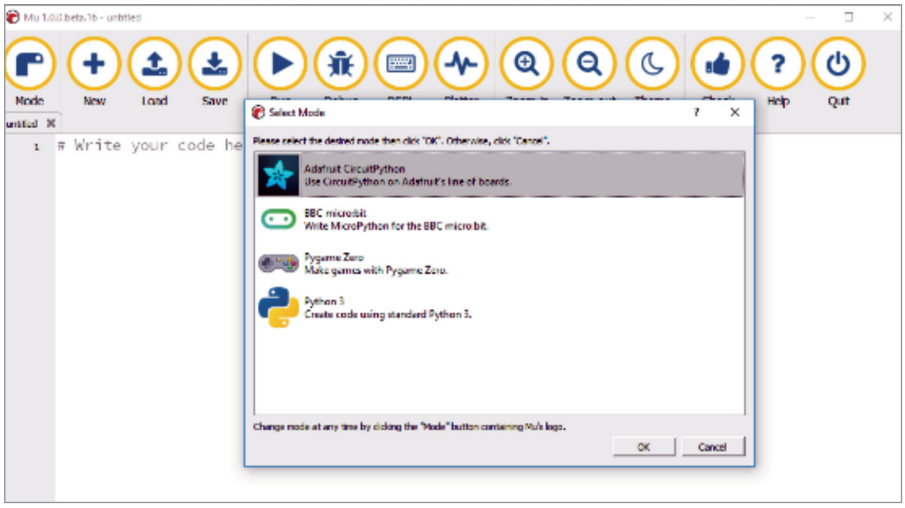


Figure 6.13: Mode selection (Python3)

Select the *Python3* mode and click OK. This will set up *mu* for programming in Python3.

eLINK

To learn more about the different mu modes, visit the following link:
futman.pub/Coding_Mu

Changing the mode

Mu will remember what mode you chose, so you only have to select it once. If you want to change the mode later, click on the Mode icon in the top left corner of the screen. Pick the mode you want from the menu and click OK.

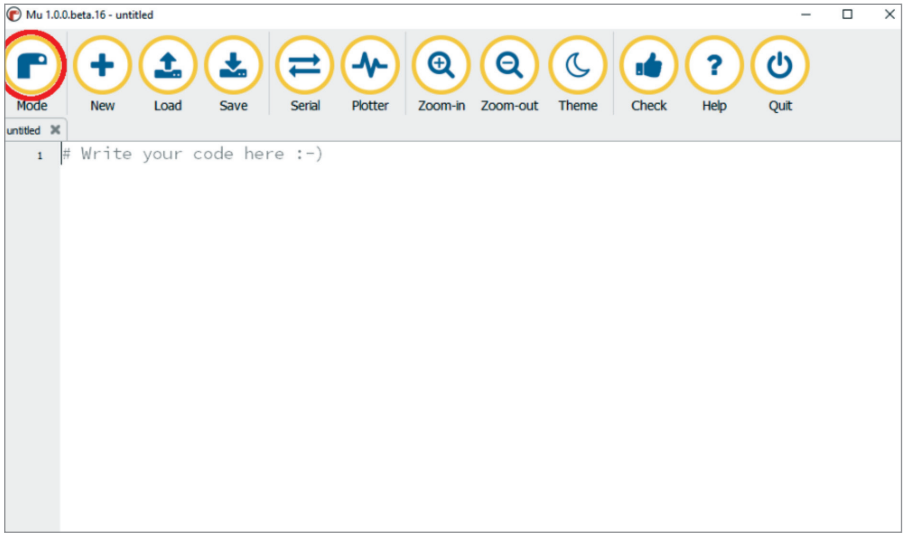


Figure 6.14: Changing the mode

[MISSING ACTUAL CODE TO ACHIEVE SPECIFIC TASK/INSTRUCTION HOW TO ACTUALLY WRITE CODE]

[NOT INCLUDED IN SYLLABUS?]

6.1.8 Deploy the code in Mu

The main area in *mu* is where you will be writing your code. For example, enter the code to create a `"Hello World"` program.

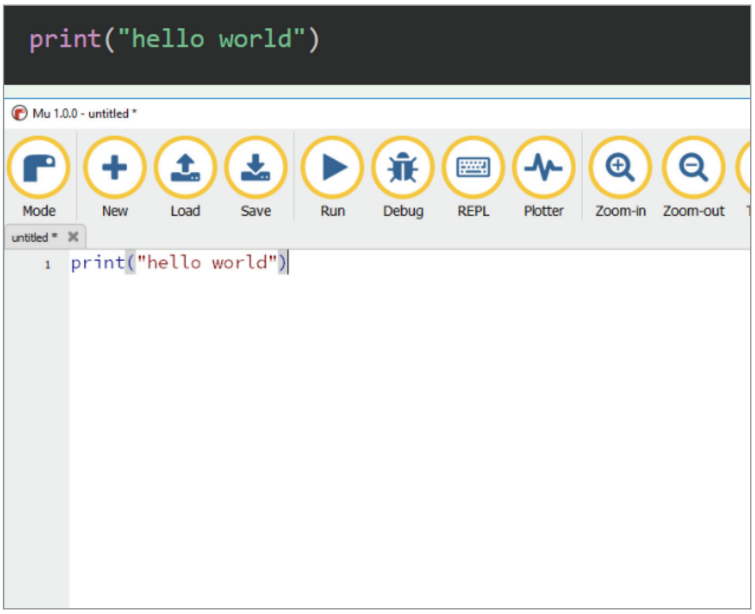


Figure 6.15: Writing "Hello world"

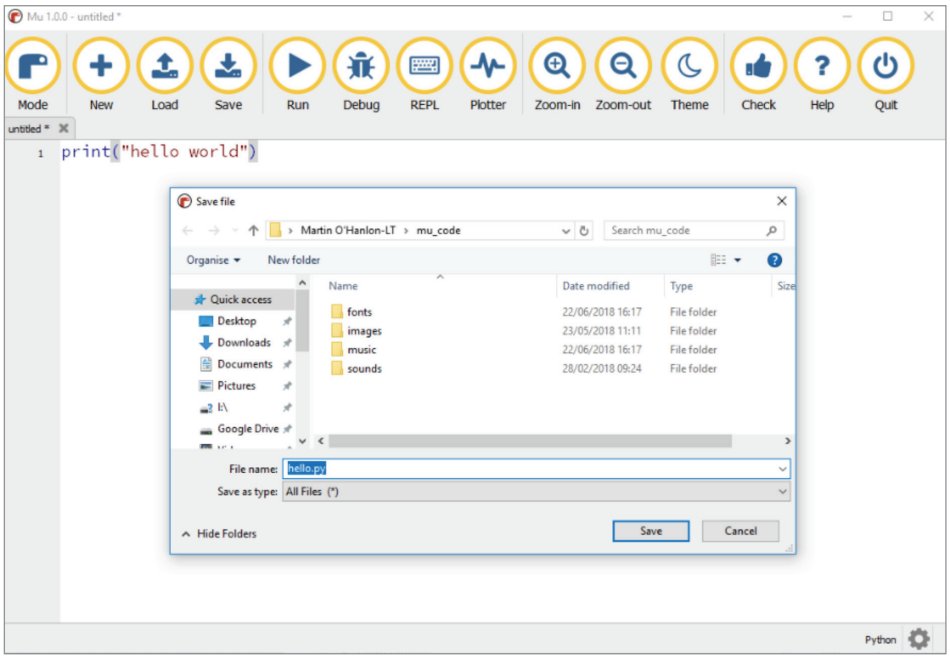


Figure 6.16: Saving your program

Click the Save icon to save your program on the local disk or external storage device. Enter the file name *Hello* and click Save in the window.

Click the Run icon to run the program and check your output, as shown on the screen below.

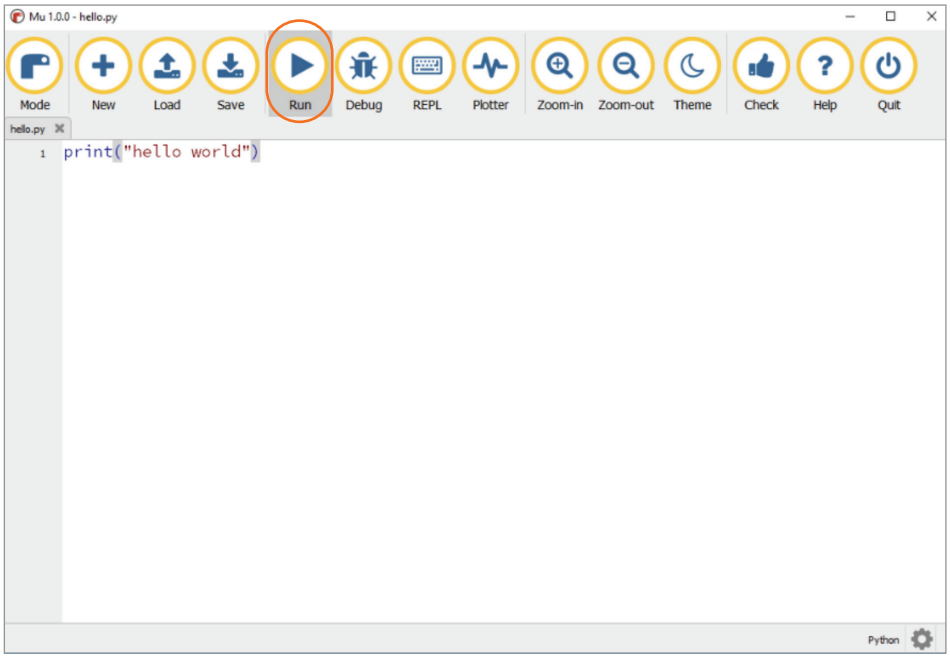


Figure 6.17: Selecting the option to run a program

Your program will run, and the message `hello world` will be displayed at the bottom of your screen, as shown below.

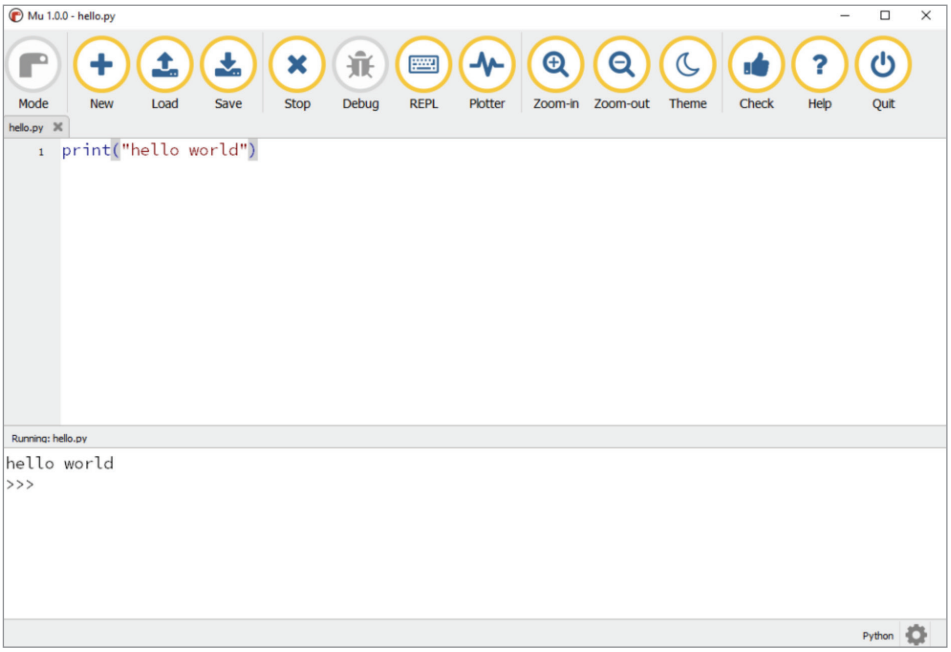


Figure 6.18: Running the program

Stop your program by clicking the `Stop` button.

Saving your code

Once you have saved your code to a file and given it a name, *mu* will automatically save it for you every few seconds, as well as every time you run it. This means that you will probably never lose any work. You can of course also use the *Save* button any time you want.

6.1.9 Test the code effectiveness based on specifications

[missing information]

6.2 Design and development of an automated artefact or prototype

Before you start designing and developing your prototype, you need to define the problem you are trying to solve. Then you can attempt to design and build a robot or artefact to solve that situation or problem.

The best approach is to write a design brief in a notebook. This will be your working document. Your notebook can be an A4 notebook or an electronic

document like MS Word or Visio. It will contain short statements which explain the problem you are trying to solve, for example:

- Exploring and designing
- Gathering information
- Identifying details of the design that must be met
- Identifying alternative and possible resolutions for the design
- Planning and designing appropriate structure that includes drawings.

6.2.1 Analyse a given problem to development of a particular robot, electromechanical artefact or prototype

After compiling the design brief, you are ready to gather information which will assist you to produce a successful design. The most important step is to decide what information you need. This will differ from project to project and depend on the specifications as well as your knowledge. A useful step is to compile a questionnaire that can help you to gather the information in a structured manner. An example is given below.

POSSIBLE QUESTIONS	POINTS TO CONSIDER
What should my prototype do, i.e. what is the practical function of the design?	The practical purpose of a design includes: ***Movement:*** How do you want your prototype to move within an environment? If it is placed in different geographical environments, will it be able to move in those environments? ***Manipulation:*** Will the prototype manipulate other objects within its environment? Should it move to manipulate more than one object? ***Energy:*** How will the prototype be powered? Intelligence: Do you want the prototype to 'think'? What does this mean? ***Sensing:*** How will the prototype sense its environment? Which visual/audio devices are needed?
What must the prototype/ artefact resemble, i.e. its form, shape, surface/texture, colours, etc.?	***Form*** and ***shape*** are critical to the design – you must consider visual quality, ergonomic, strength, stability, rigidity, durability and safety ***Surface*** and ***colour*** appropriate to the design – visually appealing, mechanical, including optical and thermal properties, etc.
What design material is appropriate?	Consider price and availability The ***properties*** of the artefact will determine the best material to use, e.g. whether you are looking for: • Toughness, strength and durability • Aesthetic qualities (determined by colour, surface texture and pattern) • Both of the above
What would be the most suitable creation technique?	Creation methods can be divided into the following categories: • Shaping and cutting • Manufacturing – the fabrication of the parts using bolts, glues, solder and screws. • Moulding – the application of a force on the material • Casting – using a mould to shape solidifying martials. The technique/process of development will be determined by the chosen material, the availability of manufacturing facilities, the skills and production cost, among other factors

POSSIBLE QUESTIONS	POINTS TO CONSIDER
What is the impact of social and environmental effects on the design?	The disposal of the product will impact humans and the environment. It is therefore your responsibility to carefully consider the potential effects of the new design in terms of health and safety factors, noise, pollution, etc.

Gathering information involves careful research regarding the type of artefact needed, its benefits/uses as well as problem-solving skills, conducting interviews and making observations. The term *specifications* refers to a detailed description of the problem to be solved. It should describe clearly what the design should accomplish.

6.2.2 List of components required

[NO INFO]

The components needed will be determined by the kit you are using as well as your design (refer to sections 6.1.1 to 6.1.4 above). Below is an example of the components used for a specific project.

[redraw and label: GPIO; Buzzer; LEDs; Resistors; Pushbutton switch
Zeb, please bring in the outside labels a bit so that total width of illustration is 125 mm. Thanks :)

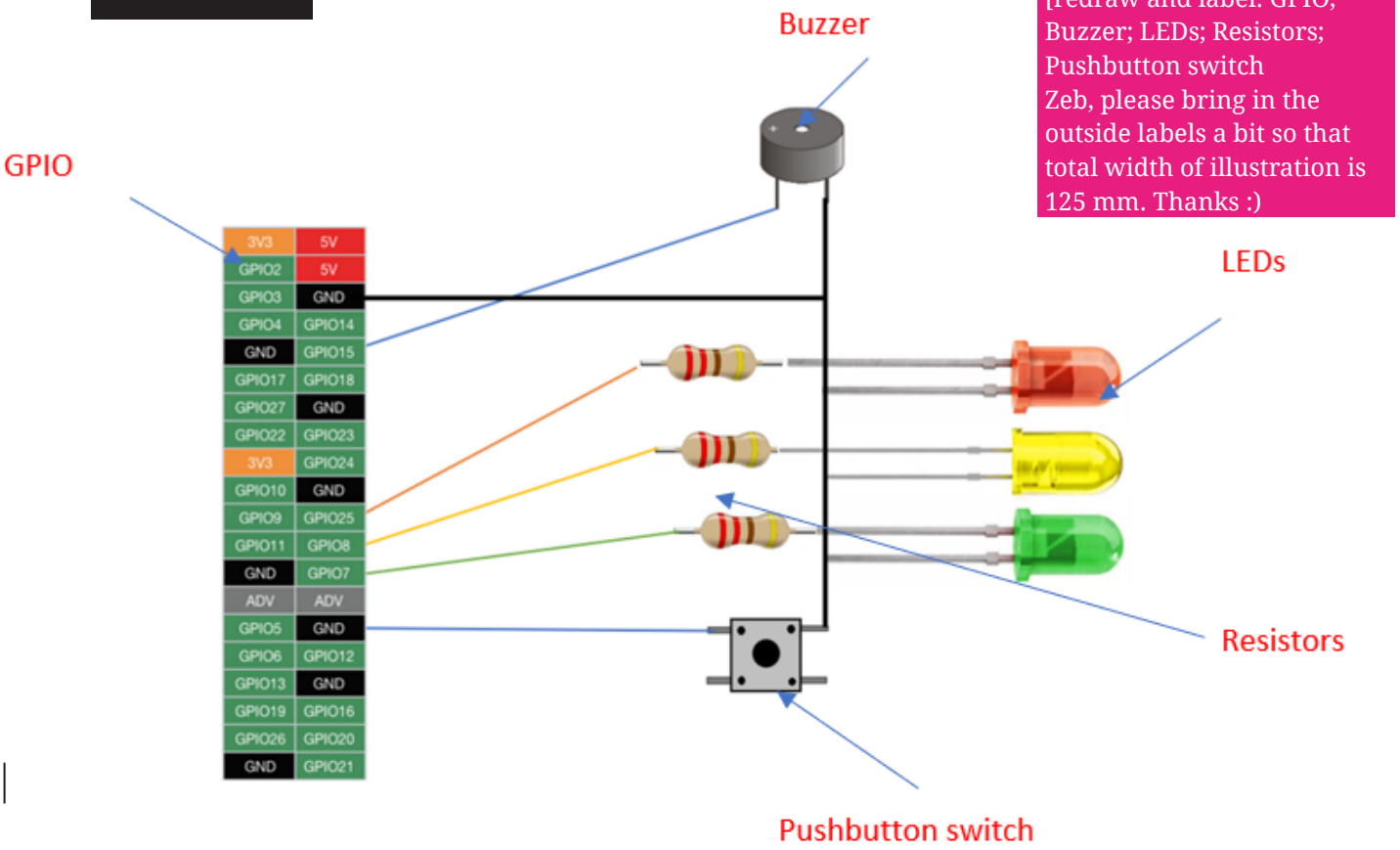


Figure 6.19: Example of components

6.2.3 Understanding the components of the design

[NO INFO]

As mentioned above, the components will vary from one project to the next. Revise section 6.1 as well as Module 4 to help you understand and interpret the components of the design. The same components pictured in section 6.2.2 are used in the figure below; however, symbols rather than actual components are used.

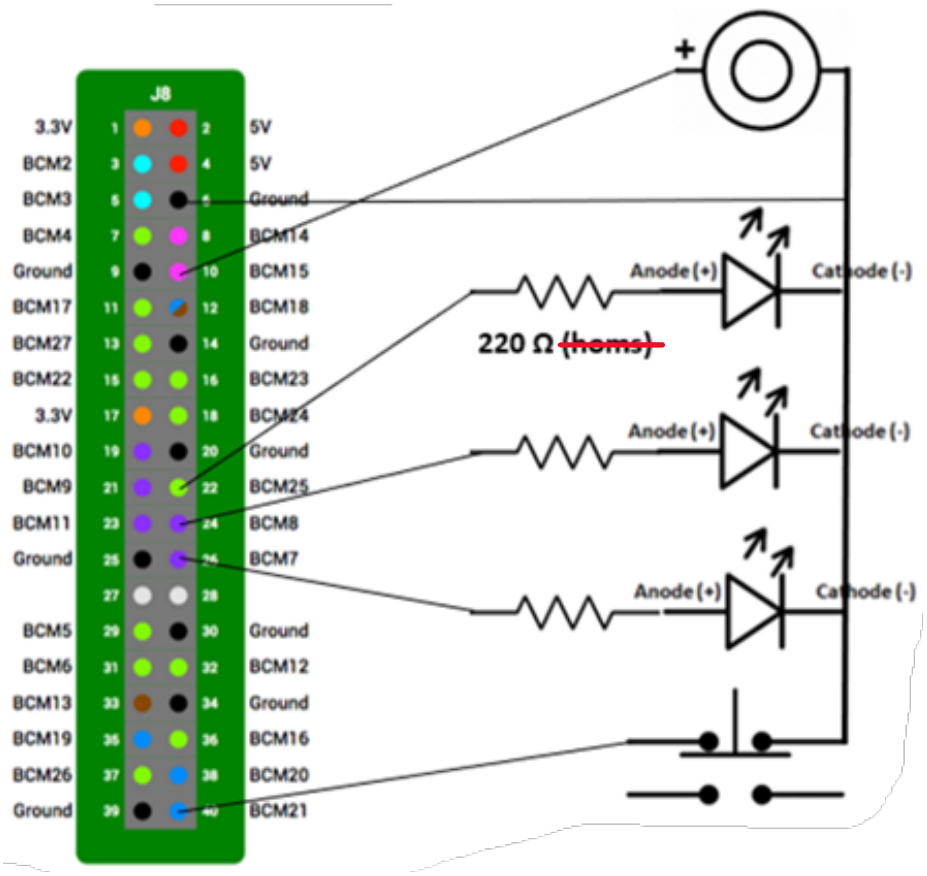


Figure 6.20: Use of symbols

6.2.4 Decomposition of the physical components needed

Lighting up an LED takes careful planning. If too much current flows through it, it will blow. To avoid this, care must be taken to limit the current by connecting a resistor in series with the LED. The long leg of an LED must be connected to the 3V3 GPIO pin and short leg to a GND pin, as illustrated below.

[Redraw; please indicate that LED is on]
Also, please make these drawings' frames just your normal frames, thanks.

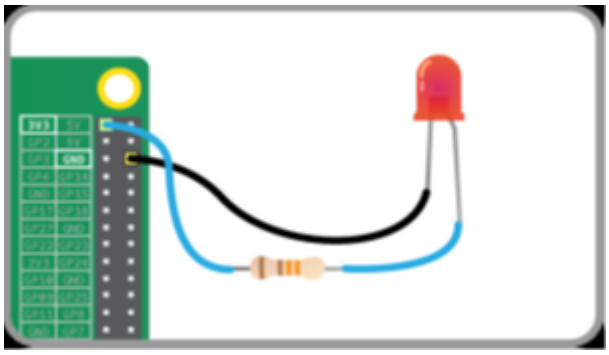


Figure 6.21: Connecting LED and resistor in series to GPIO pins

You will notice that the LED lights up and remains on because it is connected to the 3V3 pin on the Raspberry Pi which is itself always on. However, if you move the connection from the 3V3 pin to the GPIO pin 17, the LED should be off, because the jumper wire is connected to a GPIO pin which is controlled by the code that was written.

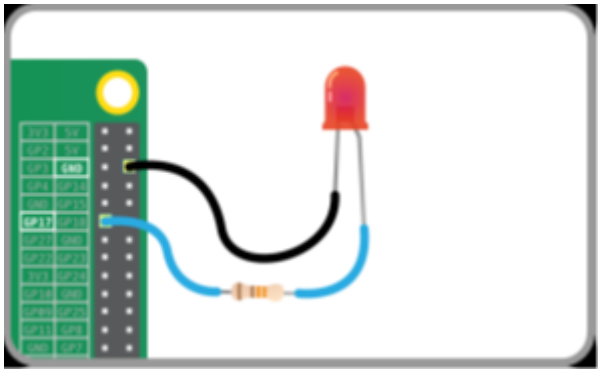


Figure 6.22: Changing the connections

6.2.5 Write out the steps for the required design

[NOTE: No clear steps provided here? Only vague references to schedule, etc.]

Ideally, you should think of at least three different ways to solve the problem before you focus on the solution. As mentioned previously, drawings and notes are essential at this stage of the design process. You can also create a prototype using LEGO in this phase. It is a good idea to take a photo of your prototype, paste it into your workbook and start making notes below the image. Once you have decided on a solution, go back over the list of requirements to ensure that each requirement is met.

The next step is to produce a working drawing. This will assist you in putting the pieces together as you start developing the prototype. You can draw the prototype on paper or use a suitable computer program.

Further steps include the following:

- Drawing up a work schedule/timetable showing the time you expect to spend on each part of the design
- Drawing up a list of the necessary tools/equipment and components needed to complete the projects
- Designing the circuit diagram – schematic
- Building the circuit/assembling components – breadboard or PCB
- Writing the Python/Mu/Scratch program
- Testing the prototype
- Troubleshooting, modifying and debugging
- Re-testing
- Presentation

6.2.6 Design the applicable circuit on paper

[Information needed – an example of a circuit drawing is needed]

6.2.7 Design the applicable circuit schematic using an appropriate software prototyping and design tool

[NOTE: There is no schematic circuit? More information/guidelines needed]

The schematic circuit below can be drawn using a copyrighted or open-source software such as PCB online drawing tools.

eLINK

To learn more about drawing your own circuit, visit the following links:
futman.pub/CircuitMaker

Using mu

Open *mu* and choose the mode you want to use (as set out in section 6.1.7). Use Python 3 if you are creating a new Python script.

[Where/what is REPL? And what should happen first?]

[NOTE: This bit of code doesn't make sense as used here – where does it fit in? Is it part of the par. above?]

You can switch an LED on and off by typing commands directly into the REPL. Click on the REPL button in the menu bar. First import the GPIO Zero library and tell the Pi which GPIO pin you are using – in this case pin 17.

```
In[1]:from gpiozero import LED
In[2]:led=LED(17)
```

Press Enter on the keyboard.
To switch on the LED, type the following and press Enter:

```
In[3]:led.on()
```

To switch it off, you can type:

```
In[4]:led.off()
```

[NOTE: Simply repeating the point in syllabus – no info provided. I think examples are needed]

Your LED should switch on and then off again.

Build the circuit

After you have designed the circuit, you can add, rearrange and connect components as needed.

[NOTE: NO prior info/ guidelines re. these functions]

Flashing a LED

With the help of the *time* library and a loop, you can make the LED flash.

Create a new file by clicking New.

[WHERE? IN MU? THAT IS THE LAST SCREEN REFERRED TO?]

Save the new file as *gpio_led.py* by clicking Save.

Enter the following code to get started:

[WHERE MUST THE CODE BE ENTERED?]

```
From gpiozero import LED
From time import sleep

led=LED(17)

while True :
led.on()
sleep(1)
led.off()
sleep(1)
```

Save the file and run the code with by clicking on Run.
The LED should be flashing on and off. To exit the program, click Stop.

Using buttons to get input

You can now control an output component (an LED). Let us connect and control an input component, a pushbutton switch.

Connect a pushbutton to a GND pin and GPIO pin 2, as shown below.

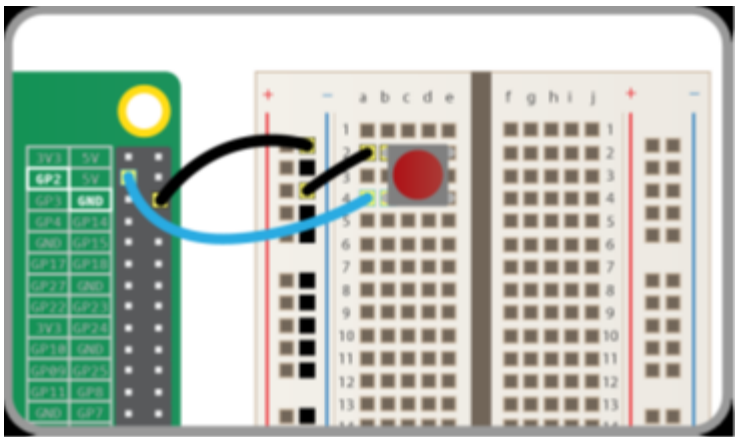


Figure 6.23: Use a button to get input

Create a new file by clicking New.

[REMEMBER TO INDICATE WHERE ABOVE]

Save the new file as *gpio_button.py* by clicking Save.

[WHERE is this option given?]

This time you must select the Button class and enter that the pushbutton is connected to pin 2. Write the following code in your new file:

```
From gpiozero import Button
Button = Button(2)
```

Now the program will respond when the button is pushed. Add the following lines:

```
button.wait_for_press()
print('You pushed me')
```

Save and run the code.
Press the button and your text will appear.

[WHAT TEXT IS THIS?]

Discussing the composition of the code

Manually controlling the LED

You can now combine the two programs written so far to control the LED using the pushbutton switch.

Create a new file by clicking `New`.
Save the new file as *gpio_control.py* by clicking `Save`.

Now write the following code:

```
From gpiozero import LED,Button
From time import sleep

led=LED(17)
button=Button(2)

button.wait_for_press()
led.on()
sleep(3)
led.off()
```

Save and run your program. When you push the button, the LED should come on for three seconds.

Making a switch

With a switch, a single press and release of the button will turn the LED on, and another press and release would turn it off again.

Modify your code as follows:

```
From gpiozero import LED,Button
From time import sleep
led=LED(17)
button=Button(2)

while True:
button.wait_for_press()
led.toggle()
sleep(0.5)
```

The *led.toggle( )* command switches the state of the LED from on to off or off to on. Since this happens in a loop, the LED will turn on and off each time the button is pressed.

It would be great if you could make the LED switch on only when the button is being held down. With GPIO Zero, that is easy. There are two options in the *Button* class called *when pressed and when released*. These commands do not block the flow of the program, so if they are placed in a loop, the program will continue to cycle indefinitely.

Modify your code as follows:

```
From gpiozero import LED,Button
From signal import pause

led=LED(17)
button=Button(2)

button.when_pressed=led.on
button.when_released=led.off

pause()
```

Save and run the program. Now, when the button is pressed, the LED will light up. It will turn off again when the button is released.

Move the applicable code to the controller

Using a buzzer

There are two main types of buzzers: active and passive. A *passive buzzer* emits a tone when a voltage is applied across it. It also requires a specific signal to generate a variety of tones. The *active buzzers* are a lot simpler to use, so these are covered here.

Connecting a buzzer

An active buzzer can be connected like an LED. However, because it is more robust, you do not need to connect a resistor to protect the buzzer.

Set up the circuit as shown below.

Normal frame pls.

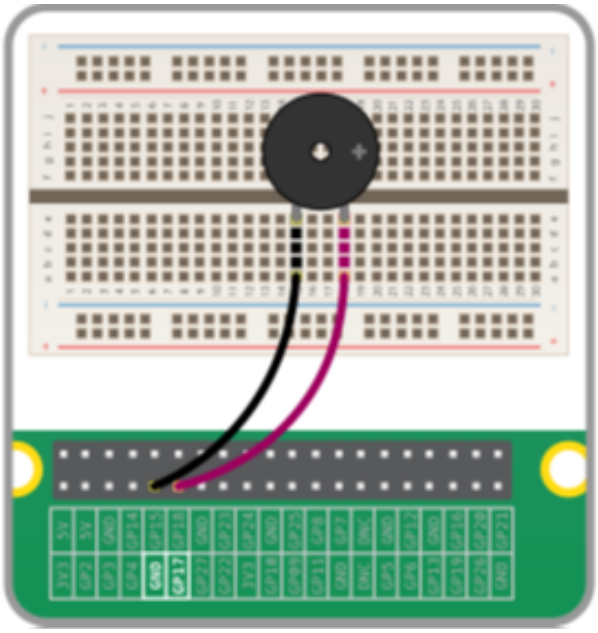


Figure 6.24: Using a buzzer

Add *Buzzer* to the from *gpiozero import...* line:

```
From gpiozero import Buzzer
From time import sleep
```

[This isn't clear]

Add a line below your creation of *button* and *lights* to add a *Buzzer* object:

```
buzzer=Buzzer(17)
```

In GPIO Zero, a buzzer works exactly like an LED, so try adding a *buzzer.on( )* and *buzzer.off( )* into your loop:

```
While True:
buzzer.on()
sleep(1)
buzzer.off()
sleep(1)
```

A buzzer has a `beep( )` method which works like an LED's blink. Try it by writing:

```
While True:
buzzer.beep()
```

6.2.8 Build or prototype the applicable circuit with applicable components

In this section, we will work on a project.

Project: Making traffic lights

Shouldn't this be a practical activity?

For this activity you will need the following:

- Breadboard
- 3 × LEDs
- 3 × resistors [OHMS????]
- 1 × pushbutton
- 1 × buzzer
- Jumper cables

Wiring

To get started, you will need to position all the components on the breadboard and connect them to the appropriate GPIO pins on the Raspberry Pi. First, you need to understand how each component is connected:

- A push button requires 1 ground pin and 1 GPIO pin
- An LED requires 1 ground pin and 1 GPIO pin, with a current-limiting resistor connected in series
- A buzzer requires 1 ground pin and 1 GPIO pin

Each component requires its own individual GPIO pin, but components can share a ground pin. We will use the breadboard to make these connections.

Connect the Raspberry Pi GPIO pins, according to the following diagram:

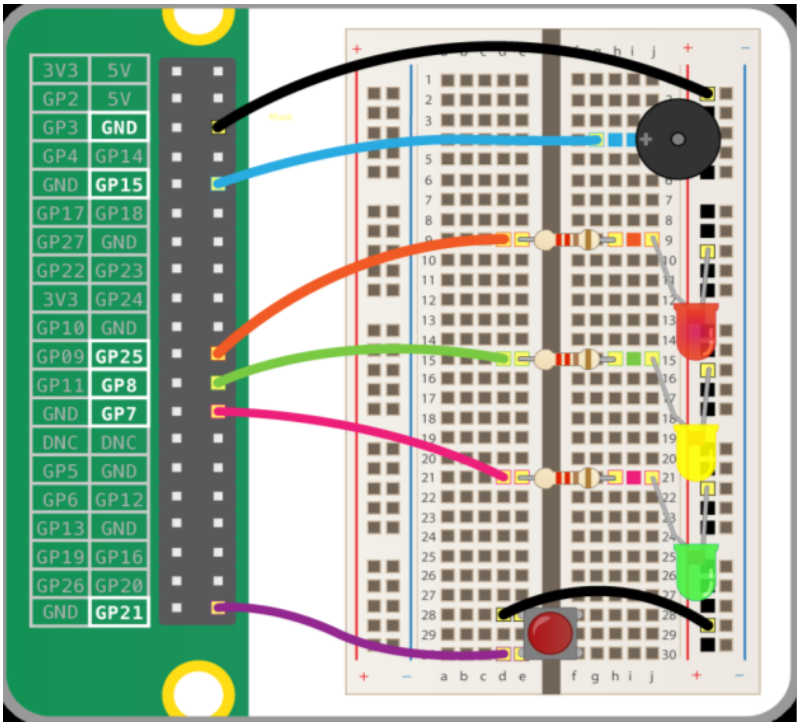


Figure 6.25: Automated prototype diagram of a traffic light

REMEMBER

The row along the vertical side of the breadboard is connected to a ground pin on the Raspberry Pi, so all the components in that row (which is used as a ground rail) are connected to ground.

Note the following table which shows to which GPIO pin each component is connected:

Component	GPIO pin
Button	*21*
Red LED	*25*
Amber LED	*8*
Green LED	*7*
Buzzer	*15*

Traffic lights sequence

Apart from controlling the whole set of lights together, you can also control each LED individually. With LEDs, a pushbutton and a buzzer, you can create your own traffic-light sequence, complete with pedestrian crossing.

Create a new design project or open an existing project

- Try some more sequences of your own, e.g. creating the full traffic lights sequence:
  Green on
  Amber on
  Red on
  Red and amber on
  Green on
  Be sure to turn the correct lights on and off at the right time, and make sure you use sleep to time the sequence perfectly.
- Try adding the button for a pedestrian crossing. The button should move the lights to red (not immediately) and give the pedestrians time to cross before moving back to green until the button is pressed again.
- Now try adding a buzzer to beep quickly to indicate that it is safe to cross, for the benefit of visually impaired pedestrians.

6.2.9 Develop, code and debug the source code for the operations of the artefact

[MISSING INFO]

6.2.10 Discuss the composition of the code

[MISSING INFO]

6.2.11 Move the applicable code to the controller

[MISSING INFO]

6.2.12 Finish and present the design and artefact

[MISSING INFO]

6.2.13 Test the artefact against a set of criteria

[Not explaining how it should be tested?]

After building and programming the prototype/artefact, you will need to test the prototype design. It is a requirement to complete a system test at several levels of the construction process. If the test shows some malfunction in a particular part of the structure or if it does not meet the set specification, you must modify your design.

Once all the programming and construction of the prototype is complete, the project needs to be subjected to another test to see whether it really does the job the design is intended to do. The evaluation needs to be written in the form of a statements setting out the strengths and weaknesses of the design. It should pronounce where the project has succeeded and where it does not achieve the desire goal set out in the specification.

The following list of question can assist you to identify and prepare the statement:

1. How well does the prototype function?
2. Does the prototype look ideal?
3. Is the artefact safe to use?
4. Did I plan my work well?
5. Did I find the creation difficult or straightforward?
6. Were the materials appropriate for the project?
7. Is the cost of the project realistic or more than expected?
8. What can be done improve in the design?