# **Module 5: Programming**

**Overview**

At the end of this unit, students will be able to:

5.1 Understand and implement basic concepts of GPIO programming

### Activity 5.1 FORMATIVE ASSESSMENT

(Individual work, written and discussion activity, self-assessed).

5.1.1 Programming is the implementation of logic to facilitate specified computing operations and functionality. Examples of programming languages include Python, Java, C# etc. (2)

5.1.2 Physical computing is the application of physical, embedded interactive systems with microcontrollers that can sense their environment and/or control outputs like lights, screens, and motors. (2)

5.1.3 Analog and Digital signals. (2)

5.1.4 Analog signal is a continuous signal which represents physical measurements. Digital signals are discrete time signals generated by digital modulation. Analog signals are denoted by sine waves while digital signals are denoted by square waves. (6)

5.1.5 GPIO stands for General-Purpose Input/Output. (1)

5.1.6 Raspberry Pi 4 has **40 pins**. (1)

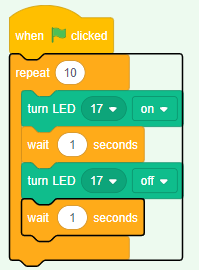
5.1.7 Lighting an LED (10)

**Items needed**

* Led
* 2 x jumper cables
* Breadboard
* 330 ohm resistor

Steps

* To control the LED using the GPIO pin you will need to start Scratch.
* Click on the Add Extension button on the bottom left corner of Scratch interface.
* Scroll down to the bottom to find three Raspberry Pi extensions
* Choose the Raspberry Pi Simple Electronics extension.
* Choose an event to start your code such as when flag clicked.
* From the Raspberry Pi Simple Electronics pallette, add turn LED 17 on
* Add a wait block followed by turn LED 17 off.
* Insert the blocks inside the repeat block to allow the LED to turn on/off 10 times. The code looks as follows:



5.1.8 An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software. (2)

5.1.9 pinout (1)

5.1.10 Temperature Convertor (10)

Code

#Entering the input

temperature = input("Input the temperature you like to convert? (e.g., 45F, 102C etc.) : ")

#Formatting input convention

degree = int(temperature[:-1])

i\_convention = temperature[-1]

#checking condition and processing

if i\_convention.upper() == "C":

result = int(round((9 \* degree) / 5 + 32))

o\_convention = "Fahrenheit"

elif i\_convention.upper() == "F":

result = int(round((degree - 32) \* 5 / 9))

o\_convention = "Celsius"

else:

print("Input proper convention.")

quit()

#displaying output

print("The temperature in", o\_convention, "is", result, "degrees.")

5.1.11 THREE different constructs used in programming

* Sequential construct
* Selection construct
* Repetition construct (2)

### Activity 5.2 FORMATIVE ASSESSMENT

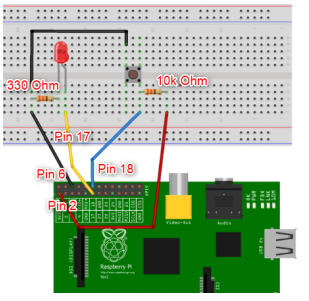
(Individual work, written and discussion activity, self-assessed).

5.2.1 Control an LED with push button

Items Needed

* Raspberry Pi
* Bread Board
* LED
* Push Button
* Resistor -330 ohm and 10K ohm
* Jumper wires - Male to Female and Male to Male

Connections



* Use GPIO pin 2 for 5V (still fine if you decide to use 3.3v)
* GPIO pin 6 for Ground to complete the circuit
* GPIO pin 17 as circuit as output- connect to the anode of LED
* GPIO 18 as input- connect to the push button
* Connect the resistors , GND and 5V

Python code

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(17, GPIO.OUT)

#LED is connected to Pin 17

GPIO.setup(18, GPIO.INPUT)

#PushButton is connected to Pin 18

try:

while True:

if GPIO.input (18) == 0:

print ("led is On")

GPIO.output (17, True)

if GPIO.input (18) == 1:

print ("led is OFF")

GPIO.output (17, False)

finally:

GPIO.cleanup ()

**Total= [20 marks]**

### Activity 5.3 SUMMATIVE ASSESSMENT

(Individual work, written and discussion activity, self-assessed).

5.3.1 FOUR factors that can be used to evaluate good code (4)

* Robustness
* Reliability
* Efficiency
* Readability

5.3.2 True (1)

5.3.3 The GPIO pins allow the Raspberry Pi to control and monitor the outside world by being connected to electronic circuits. (2)

5.3.4 The output is 12. This is eval() method parses the expression passed to this method and runs python expression (code) within the program and an addition is performed. (2)

5.3.5 Input is data entered or received by a computer. This could be temperature read from a sensor, button press signal. Output is how the computer presents the results of the process. Outputs can be returned to the user in many ways such as text on a screen, printed materials, or as sound from a speaker or LED turning on in response to a signal supplied. (4)

5.3.6 Deployment involves packaging up your application and putting it in a production environment that can run the app. (2)

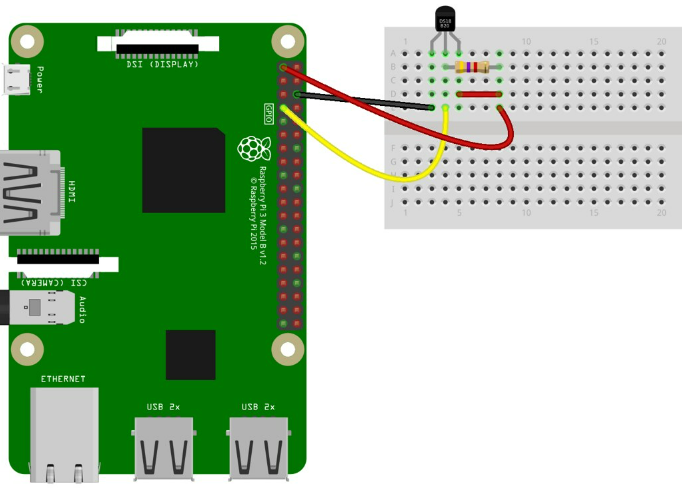
5.3.7 Temperature sensor With Python (Raspberry Pi) using DS18B20 Temperature Sensor

(15)

**Items Needed**

* A DS18B20 Temperature Sensor
* A 4.7K Ohm Resistor (Colour Code: Yellow Purple Red Gold) Or 3.3K
* A Breadboard
* 3 x Female to male jumper cables.
* 1 x Male to Male jumper cable (Optional, see the comment in the Getting Started paragraph below)

**Connection**



Use the diagram above to set up the connection.

Install the Python Library called w1thermsensor.

Enable the Interface from the Configure Raspberry Pi system screen= Enable for the 1-Wire interface.

**Python Code**

import time

from w1thermsensor import W1ThermSensor

sensor = W1ThermSensor()

while True:

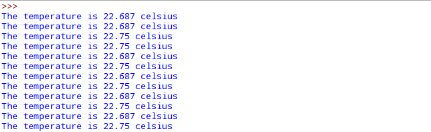
temperature = sensor.get\_temperature()

print("The temperature is %s celsius" % temperature)

time.sleep(1)

Run your code and you will see the Python Shell window pop up and start displaying temperature data

Sample output



**Total =[35 marks]**