

Lab 1: Ultrasonic Ruler

Objectives:

- Familiarise yourself with the MicroBit and Mu code editor
- Practice programming with micro-python
- Practice using the ultrasonic sensor

1. The MicroBit and Mu

1.1 The MicroBit

The BBC MicroBit is a popular pocket-sized microcontroller with a suite of built-in functionality as well as connection pins for various external modules.

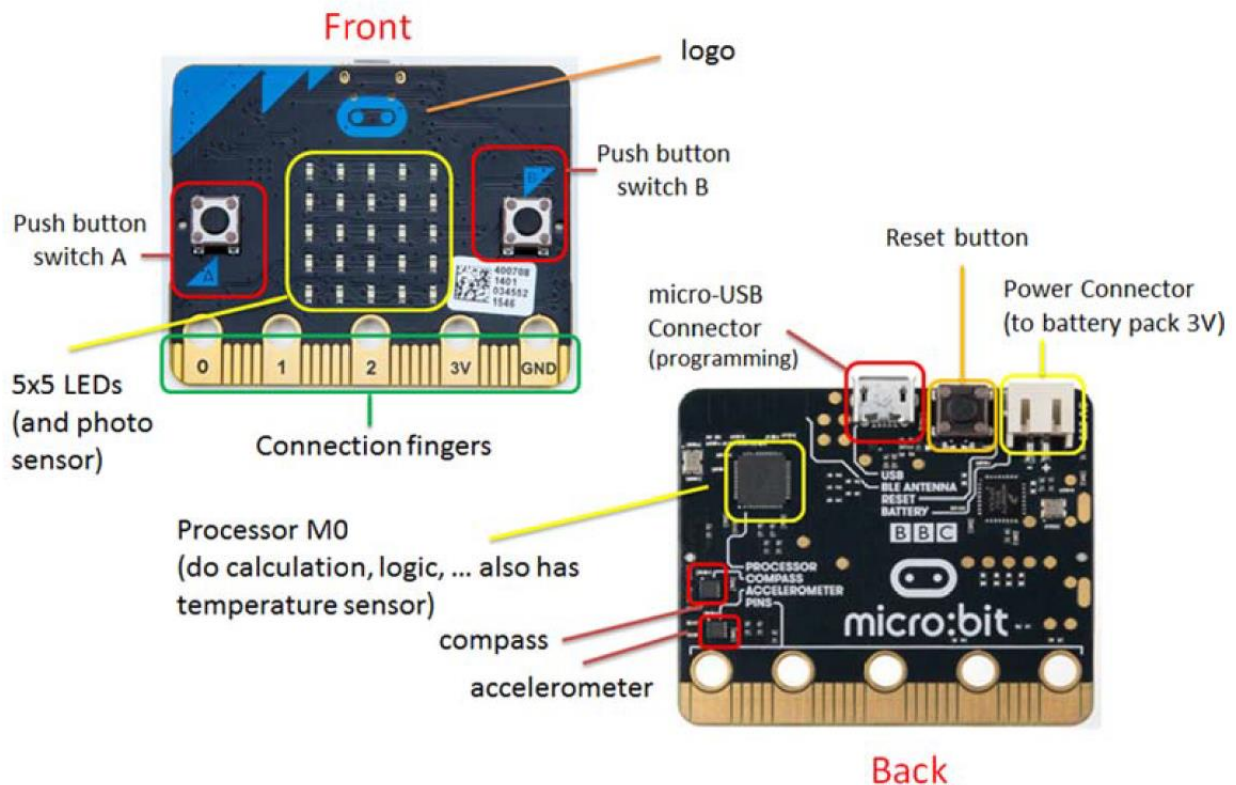


Figure 1 MicroBit components

Many creative projects can be created with the MicroBit by using several programming languages. For this lab we will be using micro-python, however it should also be possible to implement with JavaScript blocks.

1.2 Downloading the Mu Editor

1. Go to <https://codewith.mu>



Figure 2 The Mu website

2. Click on the green download button
3. Download the correct installer for your OS

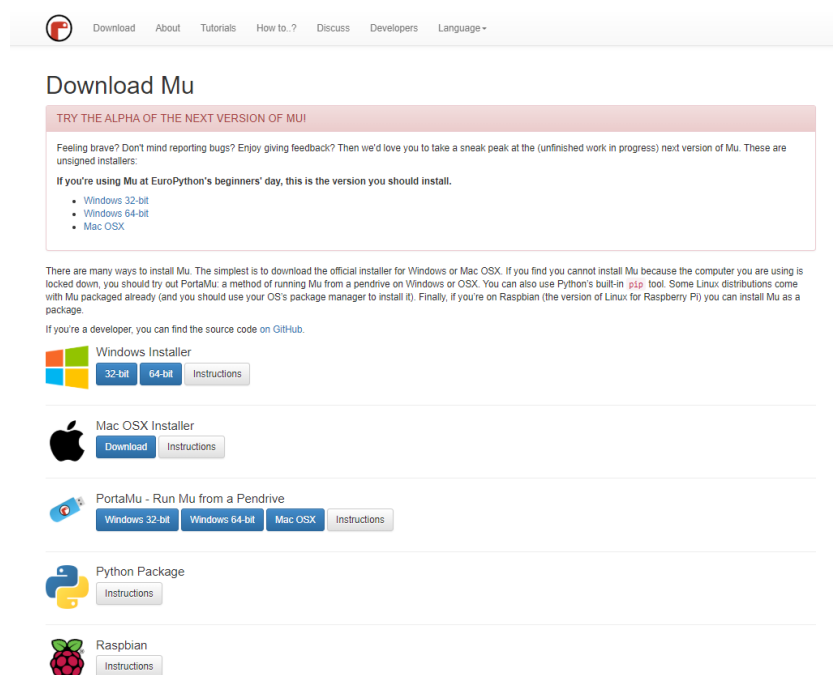


Figure 3 Mu download page

4. Follow the accompanying instructions to install Mu

1.3 Connecting the MicroBit

To connect the MicroBit to your computer and run programs, do the following:

1. Connect the MicroBit to a USB port on your computer, the yellow LED on the back should light up and the MicroBit drive should pop-up on your computer
2. Open the Mu editor and select the MicroBit mode
3. Test the connection by typing in the following code and flashing it to your MicroBit (click the flash button in the top menu bar)

```
1 from microbit import *
2
3 while True:
4     display.scroll('Hello World!')
5
6
```

Checkpoint: if the text shows up on the display, show it to a tutor.

2. Hardware and Programming

2.1 Ultrasonic Sensor

The ultrasonic sensor measures distance by emitting an ultrasonic pulse and measuring the time it takes to come back. As shown in Fig. 4, there are 4 connection pins on the sensor, the Vcc, trigger, echo and ground pins. The trigger and echo pins are located near the top right of the circuit board, whereas the Vcc and ground pins are connected for you.

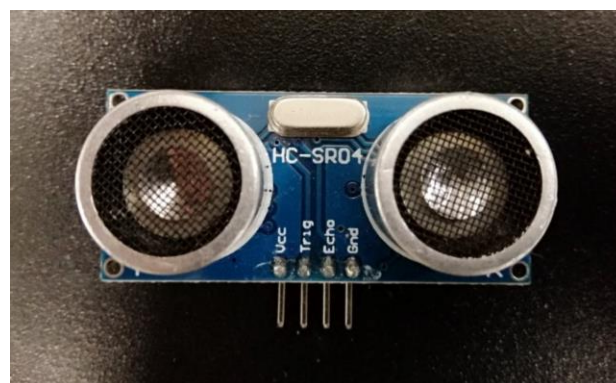


Figure 4 HC-SR04 ultrasonic sensor

The Vcc pin takes an input voltage of 5 volts, the trigger pin emits the ultrasonic pulse, and the echo pin receives the reflected pulse. Please refer to CANVAS notes and video of Week5 “Programming with MicroBit (Part I)” for more detail on the principle and some sample examples.

2.2 Hardware Connections

As shown in Fig. 5, the MicroBit has 19 pins available for connection, with different types and functions. Pin 0, 1 and 2 are the most powerful and therefore commonly used. These pins can read and write both digital and analog signals, as well as detect when they are being touched. For this lab, two pins will be connected to the ultrasonic sensor.

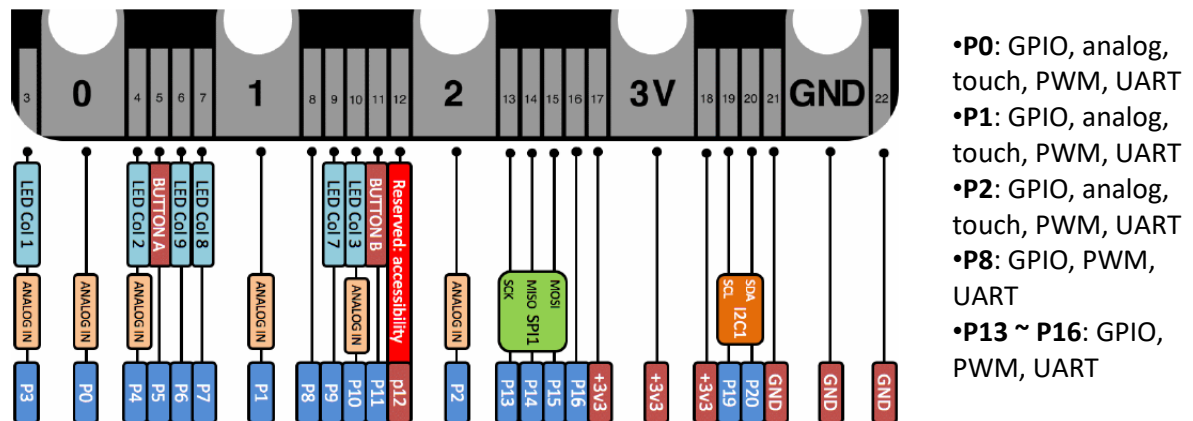


Figure 5 Input/output pins of MicroBit

Task 2.1: Choose a pin on the MicroBit which can write digital signals and connect it to the TRIG pin on the circuit board with a wire. Next, choose a pin on the MicroBit which can read digital signals and connect it to the ECHO pin.

What to do and to submit for Hardware Task 2.1?

- 1) Pin assignment between MicroBit pin and the TRIG pin of the sensor and connect the two pins with a wire physically.
- 2) Pin assignment between MicroBit pin and the ECHO pin of the sensor and connect the two pins with a wire physically.

Submit the pin assignments of (1) and (2) at CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 1

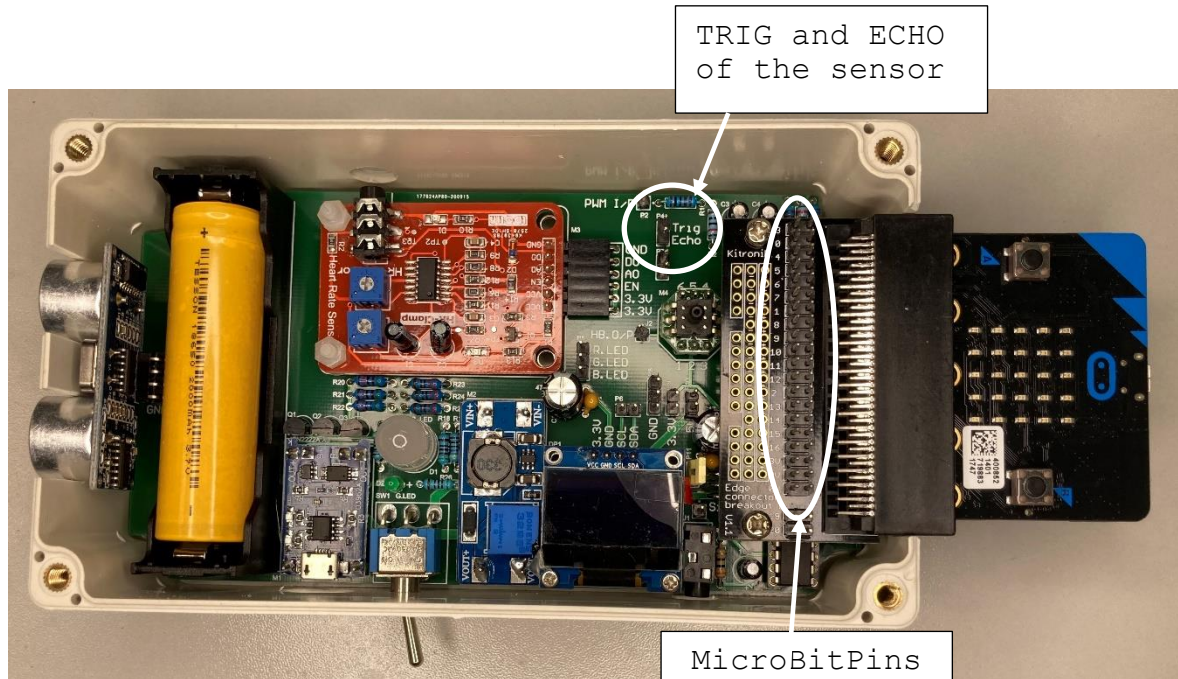


Figure 6 Lab Kit –connection of TRIG and ECHO pins

Task 2.2: Connect a 0V pin of the MicroBit to a ground (GND) pin on the PCB board, and a 3V pin of the MicroBit to the 3.3V pin on the PCB Board.

What to do for Hardware Task 2.2?

- 1) To connect 3.3V Pin of the PCB board to the MicroBit 3V pin, preferably using a red wire
- 2) To connect GND (0V) Pin of the PCB board to the MicroBit 0V pin, preferably using a black wire

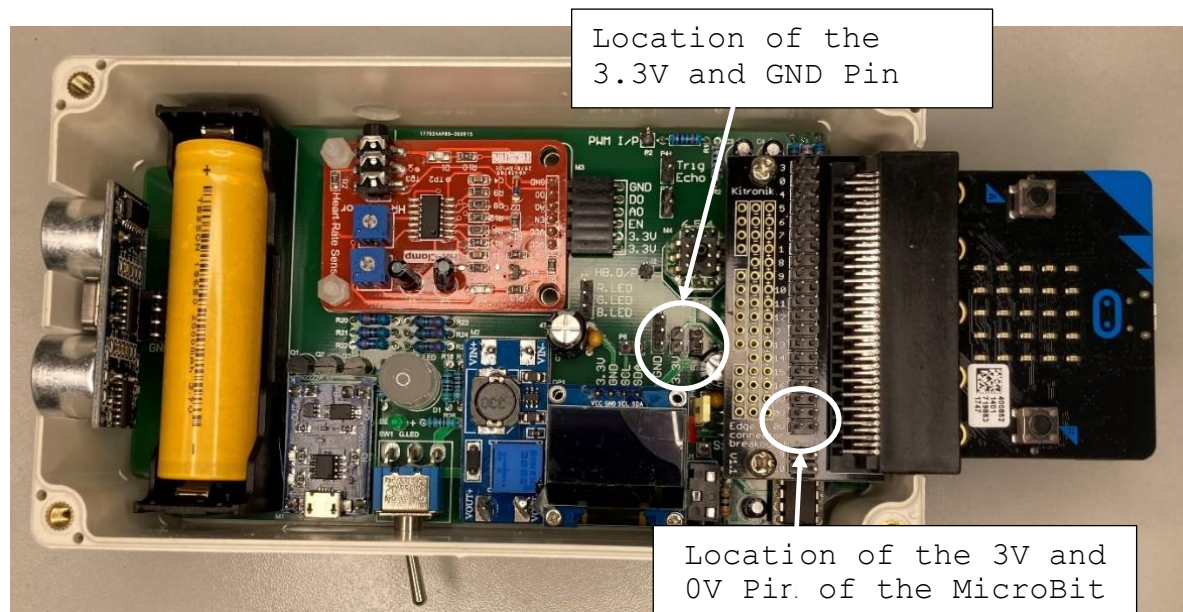


Figure 7 Lab Kit –connection of 3.3V and GND pins

2.3 Programming Tasks

Your task is to write a program for the MicroBit which utilises the ultrasonic sensor to make reasonably accurate measurements of distance over a range of 1 metre.

For access to basic MicroBit utility, you should import the microbit module using “from MicroBit import *”. This will allow you to use functions such as:

sleep(ms)	Creates a pause in your program for a given number of milliseconds
display.scroll(value)	Scrolls input on the MicroBit display, add additional argument “wait=False” to let it run in the background
button_a.was_pressed()	Returns True if the ‘a’ button has been pressed, False if it has not

In addition, refer to the following functions:

Function/method	Note	Example
write_digital(value)	Method for the digital pin class, takes values of 0 or 1	trig.write_digital(1)
read_digital()	Method for the digital pin class, returns 0 or 1	echo.read_digital()
sleep_us(us)	Function from the ‘utime’ module, takes positive integer input and delays for the given number of microseconds	from utime import sleep_us sleep_us(100)
time_pulse_us(pin,pulse_level)	Function from the ‘machine’ module, measures duration of a pulse on a given pin and pulse_level (0 or 1) in microseconds and returns it	from machine import time_pulse_us time = time_pulse_us(echo, 1)

Task 2.1: Define the TRIG and ECHO pins as instances of the correct pin class (from Task 2.1), and initialise the two pins with `write_digital` (set to low) and `read_digital`. Write your code for Task 2.1 in the box below:

What to do for Programming Task 2.1?

Sample codes: (please complete the codes value below)

```
# Assign two constant names (you decide) to represent the  
actual pins number used representatively
```

```
NAME1 = pinX (your assigned pin X number for TRIG pin)
```

```
NAME2 = pinX (your actual pin X number for ECHO pin)
```

```
# Initialize the pins with the commands
```

```
NAME1.write_digital(s1)
```

```
# set s1 = 1 for the pin output voltage equal to 3.3V,
```

```
# set s1 = 0 for the pin output voltage equal to 0V
```

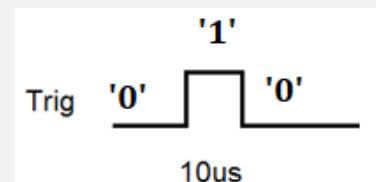
```
NAME2.read_digital()
```

Task 2.2: Write **a function** that sends 10 microsecond pulses on the trigger pin.

What to do for Programming Task 2.2?

(please complete the codes' value below)

Hints:



- 1) To generate a pulse as shown in top right hand corner from the MircoBit Trigger pin, you may apply the `write_digital(ss)` 3 times consecutively to set the pin output as '0', '1' and '0' one after another. You have to set `ss = 0` or `1` for the desire output.
- 2) To control the pulse width, you may insert the command `sleep_us(pw)` between the 2nd and the 3rd `write_digital(ss)`, where `pw` is the desire pulse width (μ s) you want.
- 3) Since it is required to write the commands within a function, we also need to use command `def` to define a function

Sample codes: (you have to choose the value for s1, s2, s3 and pw by yourself for generating a 10µm pulse)

```
#replace NAME1 with your own name which is defined the TRIG pin number
```

```
NAME1.write_digital(s1) #initialize the Trig pin to state '0'
```

```
# replace F_NAME1 with your own name for the function
```

```
def F_NAME1()
```

```
    # choose the value of s2 to set the output pin equal to '1'
    NAME1.write_digital(s2)
```

```
    # replace pw with your value to generate 10us delay
    sleep_us(pw)
```

```
    # choose the value of s2 to set the output pin equal to '0'
    NAME1.write_digital(s3)
```

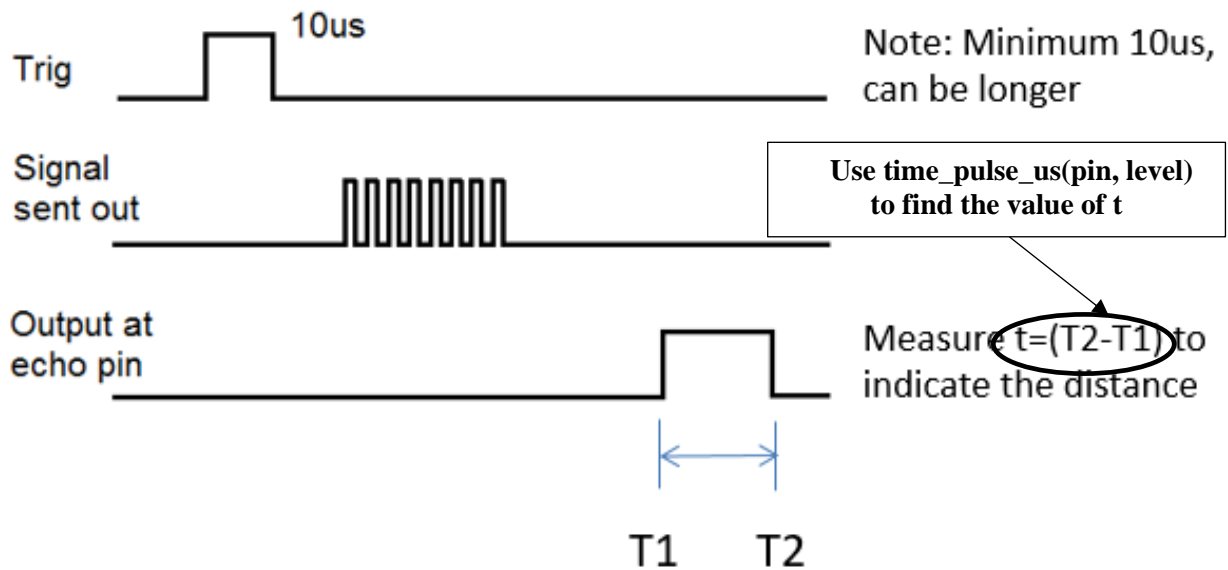
Task 2.3: Modify the function to also record the time that the echo pin is on for, take that time and return a distance in m/cm (take the speed of sound to be 346 m/s).

1. Using the function, take a measurement of the distance. (Give the unit of your answer)
Distance = _____
2. Compare your answer to the actual distance. Calculate the percentage error of your measurement.
Percentage error = _____%
3. Repeat 5 more distance chosen between 30cm to 120cm, tabulate all your results
If the difference is large, check your code for errors and take more measurements.

What to do and submit in Programming Task 2.3?

Hint:

- 1) To use the command **time_pulse_us(pin, pulse_level)** to find the pulse duration **t** of the ECHO pin as shown below.
- 2) To use command **return** to pass back the calculated distance to calling program by the function **F_Name1()**.



What to do and submit in Programming Task 2.3? (con't)

Sample codes (partial):

```
#students need to fill in other codes below
# below is the function to trigger the sensor and return the
measure distance

def F_NAME1()
# generate a trigger pulse at pin NAME1
    NAME1.write_digital(??)
    sleep_us(??)
    NAME1.write_digital(??)

#TIME is a variable to store the echo time, you replace with
other name
    TIME = time_pulse_us(ECHO pin, level)

# write a line to calculate distance DIST from TIME
# DIST is a variable to store the calculated measured distance
    DIST = ???

#pass the DIST back to the calling program
    return(DIST)
```

What to do and submit in Programming Task 2.3? (con't)

Sample codes (partial):

```
#Main program to call the ultrasonic sensor function, then  
store the return value at the variable DIST, print the value  
of DIST to the mu editor console.
```

```
DIST = F_NAME1()
```

```
#for Task 2.3 - 2.6, please do not use display.scroll()  
command.
```

```
print(DIST)
```

Submit your python codes similar to the format of my samples above to CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 2

Submit your tabulated measured data to CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 3

Task 2.4: Call your function in a while loop and then print the distance. Take readings at a regular interval by using "sleep(ms)" (e.g. "sleep(20)" for approximately 50 measurements a second), they should show up in the REPL.

Checkpoint: Show your tutor that you are able to take measurements of distance

What to do in Programming Task 2.4?

Hint:

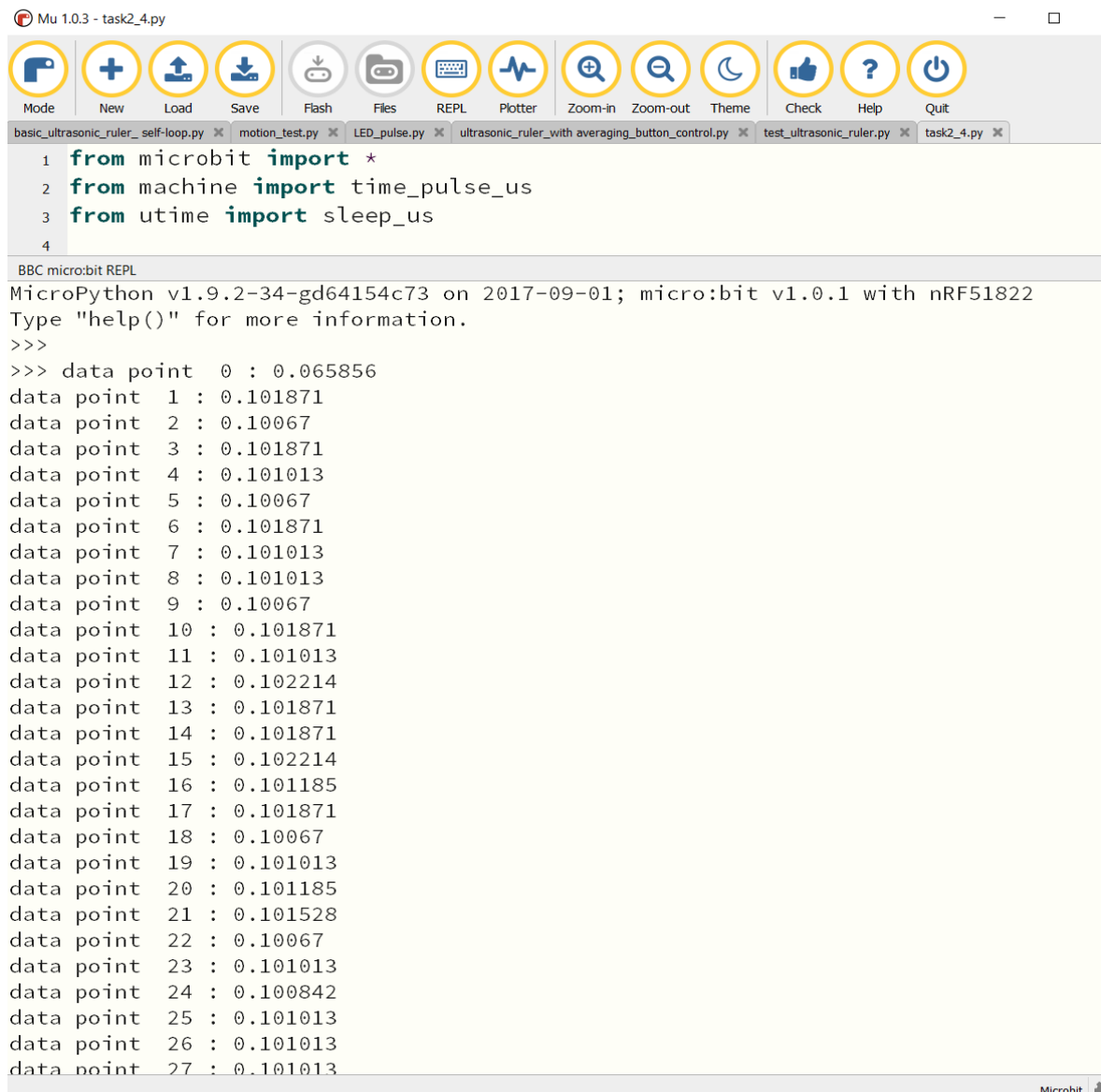
- 1) Write a **while** loop to loop 50 times; other commands is also possible, such for loop.
- 2) Within the loop should include the following commands:
 - call the function at Task 2.3
 - print the distance return at mu editor console

What to do and expected outcome in Programming Task 2.4?
(con't)

Sample codes (partial):

```
i = 1
while (i < ??):                # set value ?? for 50 loops
    i += 1
    xxxx = F_NAME1()
    print('data point ', i, ':', xxxx)
    sleep(??)                  # set value ?? for 20ms
```

Screen shot at the output of mu editor printing 50 measurements with same distance fixed. Unit is in metre.



The screenshot shows the Mu Editor interface. At the top, there's a toolbar with icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Help, and Quit. Below the toolbar, there are several open files: basic_ultrasonic_ruler_self-loop.py, motion_test.py, LED_pulse.py, ultrasonic_ruler_with_averaging_button_control.py, test_ultrasonic_ruler.py, and task2_4.py. The main editor area displays a Python script with the following code:

```
1 from microbit import *
2 from machine import time_pulse_us
3 from utime import sleep_us
4
```

Below the script, the BBC micro:bit REPL shows the output of the program. It starts with the prompt >>> and then displays 28 data points (from 0 to 27) in the format "data point X : Y", where Y is a floating-point number representing distance in metres. The values are mostly between 0.10067 and 0.101871.

```
>>> data point 0 : 0.065856
data point 1 : 0.101871
data point 2 : 0.10067
data point 3 : 0.101871
data point 4 : 0.101013
data point 5 : 0.10067
data point 6 : 0.101871
data point 7 : 0.101013
data point 8 : 0.101013
data point 9 : 0.10067
data point 10 : 0.101871
data point 11 : 0.101013
data point 12 : 0.102214
data point 13 : 0.101871
data point 14 : 0.101871
data point 15 : 0.102214
data point 16 : 0.101185
data point 17 : 0.101871
data point 18 : 0.10067
data point 19 : 0.101013
data point 20 : 0.101185
data point 21 : 0.101528
data point 22 : 0.10067
data point 23 : 0.101013
data point 24 : 0.100842
data point 25 : 0.101013
data point 26 : 0.101013
data point 27 : 0.101013
```

The bottom right corner of the window shows the Microbit logo.

Obtaining an accurate measurement:

If you plot your data from Task 2.4 (see above), you will notice that readings can fluctuate, especially during and after movement. To make an accurate metre ruler, you cannot take a single reading alone. The simple way to achieve this is by taking an average over a certain period, as well as filtering out values that are too high or negative.

Task 2.5: Since there is no built-in math module for micro-python on the MicroBit, write your own mean function that takes as input a list of values and returns the mean as a float. (Think about what will happen if the list is empty)

Write your code for Task 2.5 in the box below:

What to do and submit in Programming Task 2.5?

Hint:

- 1) To write a **function F_NAME2** which can calculate the mean of a list name **DATALIST** (you can replace with other variable names)
- 2) A list is created by placing all the items (elements) inside square brackets [], separated by commas.
Ref: <https://www.programiz.com/python-programming/list>
- 3) Suggested to use **for** or **while** loop to calculate the total sum of a list name **DATALIST**, and then divide the sum by the total number of list elements within **DATALIST**
- 4) You may use the command **len()** to determine the total number of list elements within **DATALIST** → **len(DATALIST)**

Sample Codes using for loop (partial)

```
def F_NAME2(DATALIST)
    sum = 0
    for i in DATALIST
        sum = sum + ??
    mean = sum / ???
    return (???)
```

Testing data for the function

```
datalist = [10, 8, 4, 3, 5, 7, 9, 2]
print(F_NAME2(DATALIST))
```

Output

6.0

Task 2.6: Write a new function which calls your previous functions for Task 2.3 and 2.5, saves distances which fall within an appropriate range into a list, and returns the mean of the list after it reaches a certain length.

1. Using the function, take a measurement of the distance. (Give the unit of your answer)
Distance = _____
2. Compare your answer to the actual distance. Calculate the percentage error of your measurement.
Percentage error = _____%
3. Repeat 5 more distance chosen between 30cm to 120cm, tabulate all your results
If the difference is large, check your code for errors and take more measurements.

What to do and submit in Programming Task 2.6?

Hint:

- 1) To write a **function F_NAME3** which measures the distance using function **F_NAME1()**.
- 2) For each measurement, the **F_NAME3** will check if the measured distance is between **0.3 - 1.2m**.
- 3) If yes, it will append its value in a list variable **S_DATALIST**. When the **S_DATALIST** has 10 data, **F_NAME3** will call **F_NAME2** to calculate the average distance. Finally, **F_NAME3** will then return the average distance back to the calling program
- 4) Suggested to use **for** or **while** loop to calculate the average distance

Sample Codes using for loop (partial)

```
def F_NAME3()
    S_DATALIST = []                #empty S_DATALIST
    While True:
        dist = F_NAME1()
    # check if the measured point is within range
    # and if the list contains 10 data
        if dist > ?? and dist < ?? and len(S_DATALIST) < ??:
    #append the measured distance to S_DATALIST
        S_DATALIST.append(dist)
        elif len(S_DATALIST) == ??:
            #write down command what to do
            #if 10 measurement are collected
            avg_dist = ??
            return(avg_dist)        #return value to calling program
```

What to do and submit in Programming Task 2.6? (con't)

Sample codes (partial):

#Main program to call the ultrasonic sensor function, then store the return value at the variable DIST, print the value of DIST to the mu editor console.

```
DIST = F_NAME3()
print(DIST)
```

Submit your python codes (similar to the format of my sample) above to CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 4

Submit your tabulated measured data to CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 5

Task 2.7: Call the new function in a while loop and show the distance on the MicroBit LED display at regular intervals.

What to do in Programming Task 2.7?

Hint:

- 1) Write an infinite **while** loop;
- 2) Within the loop should include the following commands:
 - To measure distance using F_NAME3()
 - To round the value of distance to 3 sig. fig.
 - To display the distance on MicroBit LED display using display.scroll()

Sample codes (partial):

```
while ??:          # set conditions for infinite loop
    xxxx = F_NAME3()
    # round xxxx to 3 sig. fig for MicroBit LED display
    display.scroll(xxxx) # show xxxx on MicroBit LED screen
    sleep(??)        # set value ?? for delay
```

Submit your complete python codes (including all import libraries, initialization of variables, define functions used, and the main program) for Task 2.7 to CANVAS "Submit your Lab 1 - Task Lists <HERE>" Question 6

Bonus challenges: revised

The following challenges are for improving your programming skills, you are encouraged to attempt at least the first one if you have already completed Task 2.7.

- 1) Press Button A to start taking one measurement and then display on MicroBit LED with unit. The display repeatedly show the measured distance until Button A is pressed again.

Hint:

- To detect whether Button A was pressed, you may use `button_a.was_pressed()`. Please refer to the link for more details: <https://microbit-micropython.readthedocs.io/en/v1.0.1/button.html>

- 2) Create a **list** to store the 5 latest measured distance in the MicroBit. The oldest data will be removed when a new data is appended to the list so that the maximum number of data stored in the list is always less than 6.

- Press Button B once to enter the display mode for showing the stored measured data in the list. The newest data will be displayed first.
- When Button B was pressed again, the next data will be displayed and etc. until all the 5 data are displayed one by one. Then the cycle will repeated again.
- At any time, press Button A once will exit this display mode. The MicroBit will then ready to take measurement again.

Hint:

- Create a list to store the 5 measured points
- Remove the oldest data in the list when new data is appended to the list.
- When Button B was pressed, it will enter a **while loop** to display the data in list one by one (for each Button B was pressed).
- When Button A was pressed, it will exit this while loop so that the program will be ready to take measurement again.

- 3) Button A + B together once will change the measuring unit from one unit to another unit from 'm' → 'cm' → 'feet' → 'inch' → 'm' and vice verse.

- When Button A was pressed to take a new measurement, MicroBit will display the measured distance with current selected unit. For example, if the current unit is 'm', the MicroBit will display 1.5m. For the same measured distance in 'cm', the Microbit will display 150 cm, and etc.

Hint:

- Create a variable to store a number representing each kind of units. For example, 1- 'm', 2- 'cm', and etc.
- When taking a new measurement, the program should use a constant for the speed of the sound based on the unit used. For example, if 'm' is used, speed of sound should be 346m/s, if 'cm' is used, the speed of sound should be 34600 cm/s, and etc.
- When displaying the measured distance, the program should show the measured distance together with the correct unit.

3. Appendix

3.1 Finding the Correct Mu version

Windows comes in 32- and 64-bit versions, and the Mu editor has two corresponding versions available for downloading. If you are unsure which version your computer uses, go to ‘Settings > System > About’ and it will be listed as the ‘System type’ as shown in the image below.

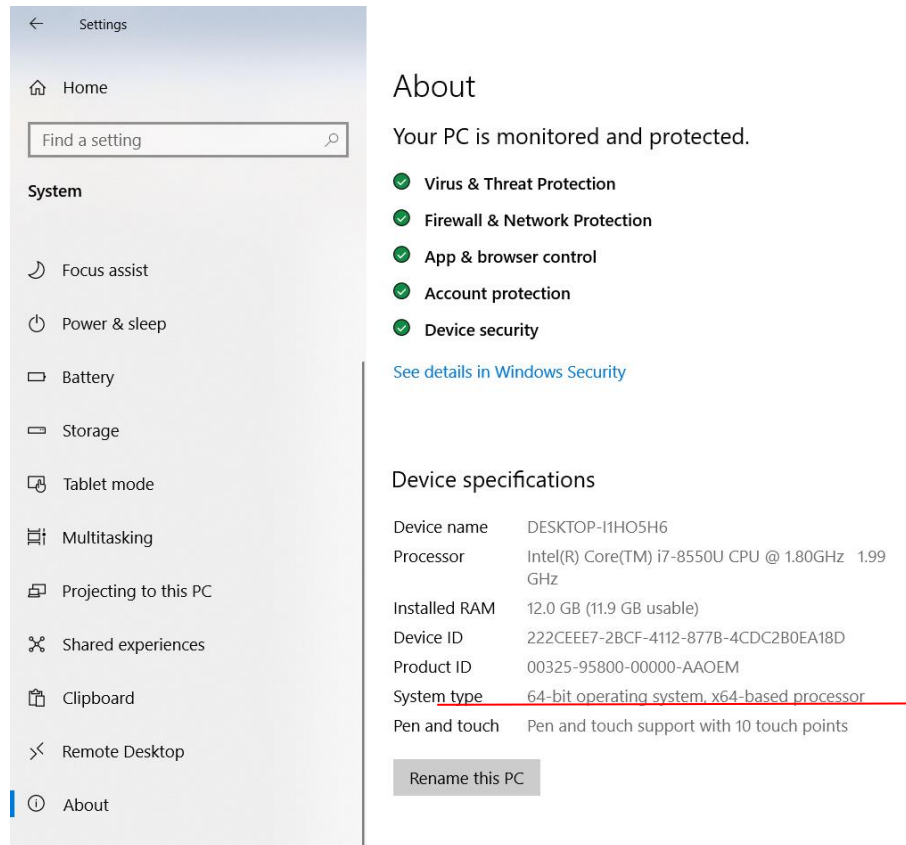


Figure 8 Finding system type on Windows 10

3.2 Basic Mu Functions

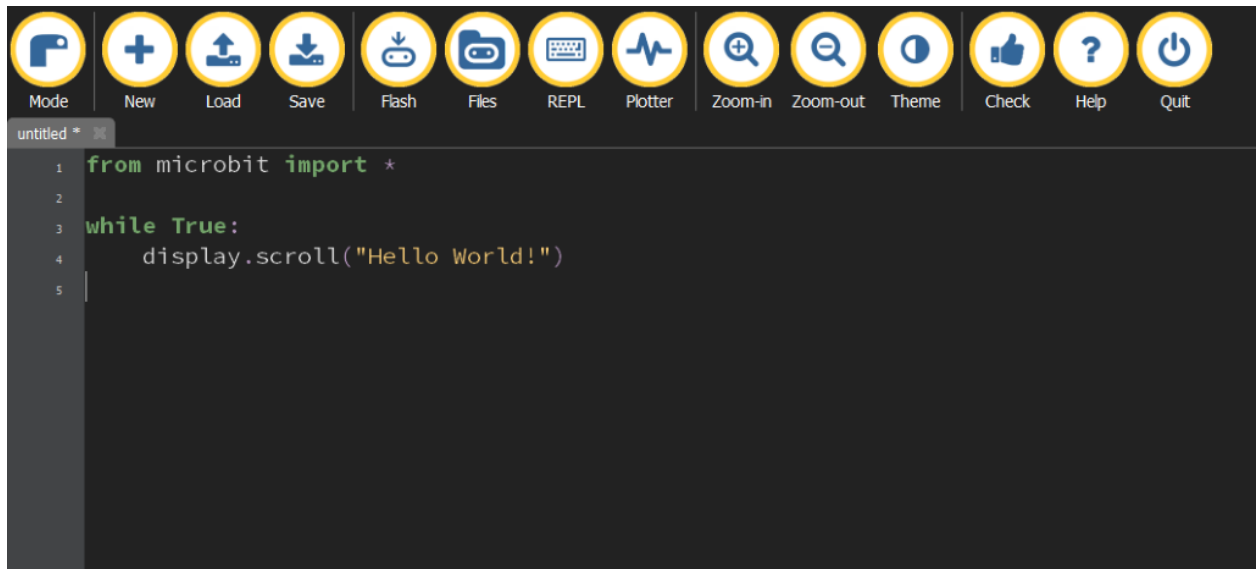


Figure 9 Mu editor

To access the following utility, press the ‘Mode’ button in the top left corner and select ‘BBC MicroBit’

- ‘Flash’:
 - Can be used to copy your code onto a connected MicroBit, code will start running automatically (note that flashing might cause the mu editor to stop responding temporarily, should start working again once flashing is complete, therefore it is a good idea to save your code before flashing)
 - During flashing, the yellow LED on the back of the MicroBit will begin flashing rapidly
- ‘Files’:
 - Displays files stored on the MicroBit and in the ‘mu_code’ folder located on the pc (default location is at “C:\Users\<username>\mu_code”)
 - Files can be transferred bidirectionally by ‘drag-and-drop’
 - ‘main.py’ is the python file which your code is copied onto during flashing and runs automatically by default
- ‘REPL’:
 - Can be used to live-code on the MicroBit
 - Displays incoming information from the MicroBit (requires using the print statement in your code)
 - Shows error statements if they occur

- To use the REPL, flash your code onto the MicroBit, open the REPL, then press the reset button on the MicroBit
- ‘Plotter’:
 - Plots incoming serial data as a line chart
 - Data must be in a tuple, i.e. `print((data_1, data_2))`
 - To plot data continuously, place print statement in a while loop (include a pause using the sleep function to avoid flooding the plotter)
- ‘Check’:
 - Checks for errors in your code, such as syntax errors, typos, etc.
 - ‘Missing whitespace’ can be ignored as it doesn’t affect code operation

3.3 Troubleshooting Connection Issues for the MicroBit

If you are having problems connecting the MicroBit to your computer, there are several things you can try:

1. Unplug and reconnect the MicroBit a few times
2. Try a different USB cable (must be one that supports data transfer)
3. Try a different USB port or computer (Note: make sure the USB port is not a fast-charging one, the MicroBit can be damaged if connected to one)
2. Reinstall MBED driver by following these instructions: <https://support.microbit.org/support/solutions/articles/19000089574-beta-testing-mbed-driver-doesn-t-work-with-windows>
5. Update the firmware on the MicroBit to the latest version: <https://microbit.org/get-started/user-guide/firmware/>
6. If all the above fails, use another MicroBit

3.4 Ultrasonic Sensor

The ultrasonic sensor (HC-SR04) measures distance by transmitting a short pulse from the trigger pin and timing the delay until the reflected signal is received at the echo pin.

As shown in the image below, the echo pin is set to high almost immediately after the pulse is sent on the trigger pin, and is turned off when a signal is received. Therefore, the duration of which the echo pin is turned on corresponds to the time it takes for the pulse to travel to the object and back. This time can then be used to calculate the distance to a high degree of accuracy.

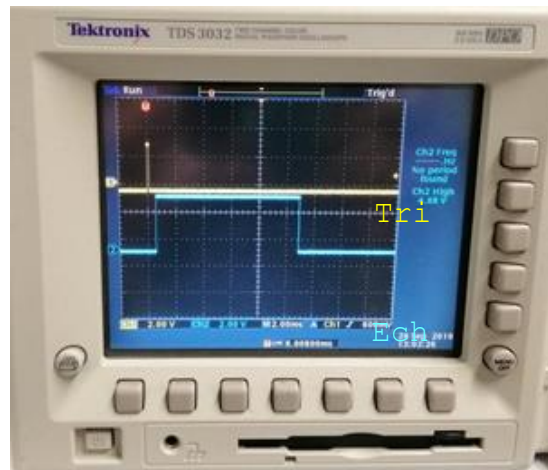


Figure 10 Ultrasonic sensor trigger and echo pulses

The short delay between the end of the trigger pulse and the start of the echo pulse means the ultrasonic sensor has a small deadzone of 2 cm. It also has an effective range of 4.5 m.

3.5 PCB Component Layout of LabKit

Fig. 10 shows the components layout of the Lab Kit with the edge connectors being removed.

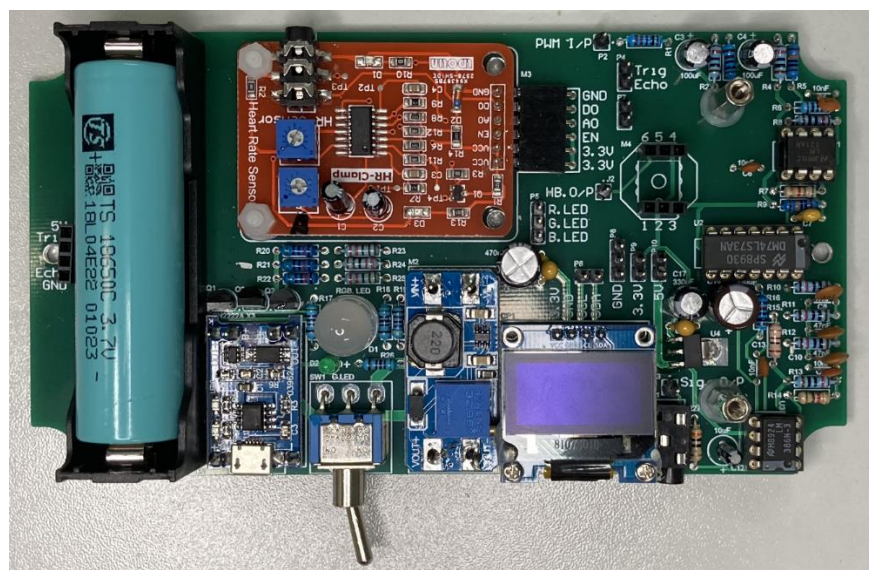


Figure 10 Components Layout on PCB with edge connectors being removed