Embedded Systems

Raspberry Pi, Python and I^{2}C

1. Collect a kit

2. Research the sensor

a. What does it do?

b. What is the power supply voltage? Some of the modules have power supplies in addition to the sensor chip so look for the documentation for the module.

c. What is the control flow for the sensor?

  i. Does it need enabling or configuring before use?

  ii. Does it measure automatically or on demand?

  iii. How do you configure it (if applicable)?

  iv. How do you request a measurement (if applicable)?

  v. How do you read back the result?

  vi. What conversion is needed to convert the result into something meaningful?

3. Establish communication with your Raspberry Pi

In this guide (only type the command):

```
raspberrypi:~$ command      #Linux prompt command on the Pi
host:~$ command             #Linux/macOS prompt command on your laptop
>>> command                 #Python prompt command
```

a. Connect your FTDI USB to Serial cable to your laptop and find the port name. Don't connect the cable to the Raspberry Pi yet

  i. In Windows:

    1. Open Device Manager and look under 'Ports' for an entry like 'USB Serial Port (COMn)'

    2. The port name is the part COMn

  ii. In MAC or Linux:

    3. Open a command prompt and type

```
host:~$ ls /dev/{tty,cu}.*
```

    4. Try the command with and without the cable connected and look for a line which only appears when it is connected. That is the name of your serial port.

  iii. If the system does not recognise the USB device, install the serial port driver from here: https://www.ftdichip.com/Drivers/VCP.htm

b. Open a terminal over the serial port

  i. On Windows, install a terminal client like PuTTY

    5. Check the box 'Serial'

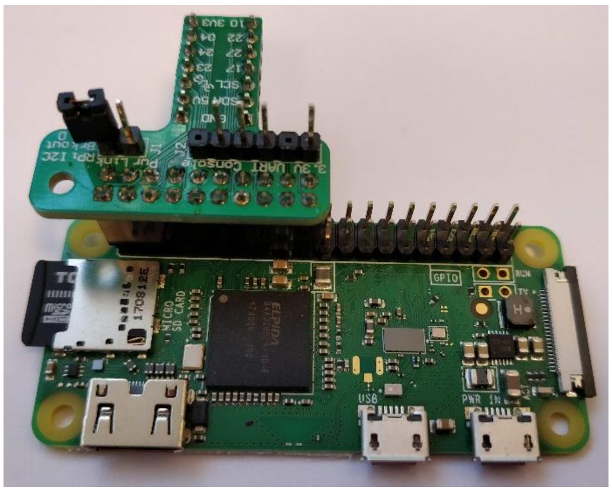
    6. Enter the serial port name under 'Serial Line'

    7. Set the speed to 115200

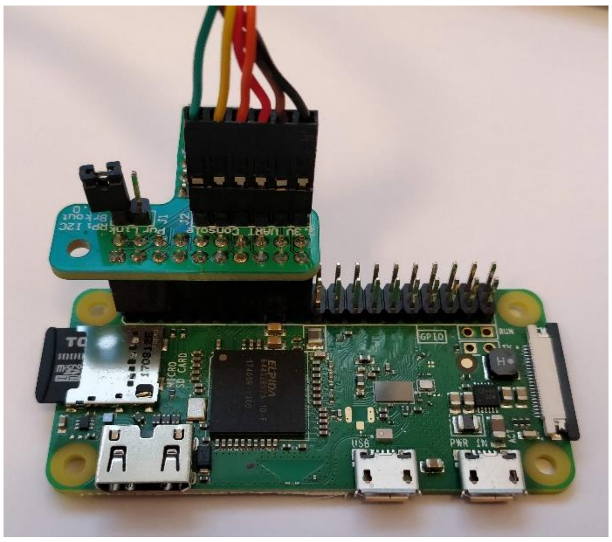
  ii. On MAC or Linux, you can use `screen` from the command line. This example is where `/dev/ttyS0` is the serial port:

```
host:~$ screen /dev/ttyS0 115200
```

c. Connect and boot the Raspberry Pi
   i. Insert the SD card into the Raspberry Pi
   ii. Connect the breakout PCB to the Raspberry Pi.



   iii. Connect the FTDI cable to the breakout PCB in the correct orientation



   iv. Bridge the power link to power up the Raspberry Pi. This supplies power from the USB cable.

d. Log in to the Raspberry Pi
   i. Watch your serial terminal to see boot messages from the Raspberry Pi. It may be blank for a little while on the first boot because it resizes the SD Card partition.
   ii. Log in with the username `pi` and the password `raspberry`

Take care with connections to the Raspberry Pi header. The USB cable provides 5V and the Raspberry Pi I/O pins can only tolerate 3.3V. Always check the positioning and orientation of the breakout board and USB connector before connecting the power link.

4. Set up the Raspberry Pi

a. Connect to WiFi

i. On the Raspberry Pi console, create a hash of your College password so it can't be easily read from the SD Card. The hash is a 32 digit hexadecimal number.

```
raspberrypi:~$ echo -n plaintext_password_here | iconv -t utf16le
        | openssl md4
```

ii. Clear the command history to remove your password from it. Make sure you close the serial console when you are finished so it can't be read by scrolling back.

```
raspberrypi:~$ history -c
```

iii. Edit the wpa_supplicant file to add the network details

```
raspberrypi:~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

iv. Add the following lines to the end of the file, replacing *uuu* with your username and *ppp* with password hash you just calculated. Save the file with Ctrl+O and exit with Ctrl+C.

```
network={
        ssid="Imperial-WPA"
        scan_ssid=1
        key_mgmt=WPA-EAP
        identity="uuu"
        password=hash:ppp
        eap=PEAP
        phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
  }
```

You can add additional `network={}` stanzas to the file to connect to other networks. The device will connect to whichever is available. Search online to find the correct syntax for other types of WiFi network

v. Restart networking

```
raspberrypi:~$ sudo systemctl restart networking.service
```

vi. Check you have a network connection

```
raspberrypi:~$ ping google.com
```

Hashing the password is not completely secure. It just prevents someone from reading your password by quickly viewing the wpa_supplicant.conf file. Two problems remain:

1. The hash gives anyone the ability to log into a WiFi network that requires your password.
2. The hash could be broken to find your password – though this is not easy. https://www.lse.epita.fr/data/2012-lse-summer-week/wifi.pdf

Be careful with access to the SD card and change your College password if you lose it.

b. Enable the SSH server so you can log in and transfer files via the network
   i. Change the Raspberry Pi password to prevent anyone else from logging in

```
raspberrypi:~$ passwd
```

   ii. Run the configuration utility and use the menu system to enable SSH under 'Interfacing Options'

```
raspberrypi:~$ sudo raspi-config
```

   iii. Find the IP address for your Raspberry Pi. Look in data in the wlan0 section

```
raspberrypi:~$ ip addr
```

   iv. Log in to your Raspberry Pi from your computer over the network using SSH
      8. In Windows, use PuTTY, or another SSH client
      9. In Mac, Linux or Windows with openSSH installed, use the command line (example IP address shown):

```
host:~$ ssh pi@146.169.152.34
```

You should be able to connect to the Raspberry Pi from any computer on the same network, depending on the structure of the network. In general, you won't be able to reach it via the internet.

The IP address is likely to change if you restart or reconnect the Raspberry Pi. You may be able to fix it if you have control over your DHCP server, but you can't on the College network.

   v. Copy a file (just a text file to test) from your computer to the Raspberry Pi
      10. In Windows, use WinSCP or a similar client
      11. In Mac, Linux or Windows with openSSH installed, use the command line:

```
host:~$ scp test.txt pi@146.169.152.34:~/test.txt
```

5. Establish communication with your sensor

a. Power down the Raspberry Pi
   i. Shut down the operating system. This is not essential, but it does reduce the chance of file corruption and should be done before any power down.

```
raspberrypi:~$ sudo halt
```

   ii. Remove the power link

b. Plug the Raspberry Pi breakout adapter and sensor module into different locations on the breadboard

c. Wire up the sensor module
   i. Check the documentation for the sensor module (the PCB, not the chip itself) to find the power requirements.
   ii. Use 3.3V if possible to reduce the chance of damage
   iii. Wire up the power supply
   iv. Wire up the I2C data and clock lines (SDA and SCL)

d. Power the Raspberry Pi again and log in via serial or SSH

e. Enable I2C and reboot

```
raspberrypi:~$ sudo raspi-config
raspberrypi:~$ sudo reboot
```

f. Search the I2C bus for your sensor module.

```
raspberrypi:~$ sudo i2cdetect -y 1
```

The command shows a map of all the I2C addresses. You should see an entry at the address you expect from reading the sensor documentation. Note that some devices have an address that is configurable by making connections to certain pins.

6. Set up Python

a. Install pip, the python package manager, and the smbus and gpiozero modules

```
raspberrypi:~$ sudo apt-get install python3-pip python3-smbus python3-
      gpiozero
```

b. Use the interactive python terminal to get data from your sensor using the library functions shown in the lecture

```
raspberrypi:~$ python3
```

c. Convert the bytes of data returned from the sensor into useful numerical values
d. Write the functions into a python script for future use
   i. On the Raspberry Pi, use the nano text editor, or install and learn to use `vim`
   ii. On your computer, use your favourite text editor and transfer the files with `scp`

> This guide covers the simplest methods of developing on the Raspberry Pi. But it is not very efficient to edit files directly on the Pi and copying from your computer is slow and can cause mistakes with file versions.
>
> Alternatives you can investigate include:
>
> - Set up VNC to give you a graphical interface to the Pi
> - Use `rsync` to synchronise files between the Pi and your development machine
> - Use `git` to synchronise files between the Pi and your development machine and manage contributions from multiple team members
> - Install and run a Jupyter notebook server on the Raspberry Pi, which can be accessed on a computer via a web browser