UNIVERSITY OF EAST ANGLIA

School of Computing Sciences

E3I Club Short Course — Raspberry Pi Interfacing

INTRODUCTION TO THE PI — Getting Started

1 Introduction

In this lab session, we are going to work through some basic example projects with the Raspberry Pi to help familiarize you with the Raspbian operating system, the architecture of the Pi and the possibilities to interact with external peripherals and sensors using the general purpose input/output (GPIO) ports.

To assist in prototyping different configurations of electronic components and sensors with the Raspberry Pi, you will be using what is known as a breadboard. A breadboard is composed of a number of rows of holes, where the holes on each row are electrically connected. This allows you to connect together a circuit easily without the need to fabricate any custom hardware.

An example of a circuit on a breadboard is shown in figure 1. In this circuit the positive terminal of a 9 volt battery is wired into a hole on row 17 of the breadboard. One end of a 270 Ohms resistor is also wired into a hole on row 17. As these connection are both made on row 17, an electrical connection is made between these components. The other end of the resistor is connected into row 21, as is the positive leg of a light emitting diode (LED). Finally a connection is made from the negative terminal of the battery to a hole on row 22, where the negative leg of the LED has been connected.

To assist you in connecting your breadboard circuits to the Raspberry Pi, each of you have been provided with a Raspberry Pi GPIO breakout board. These come in kit form and require some basic soldering to assemble. Your first exercise will involve soldering the pins onto these so that you can use them in the following exercises.

2 Raspberry Pi Basics

2.1 Connecting the Peripherals

While it is possible to connect to the Raspberry Pi remotely from another, network connected computer, in this session, we will be using the Raspberry PI directly with a screen, keyboard and mouse. As such you will need to start by connecting the following items:

- SD Card
- USB Keyboard & Mouse
- HDMI Cable
- Ethernet Network Cable

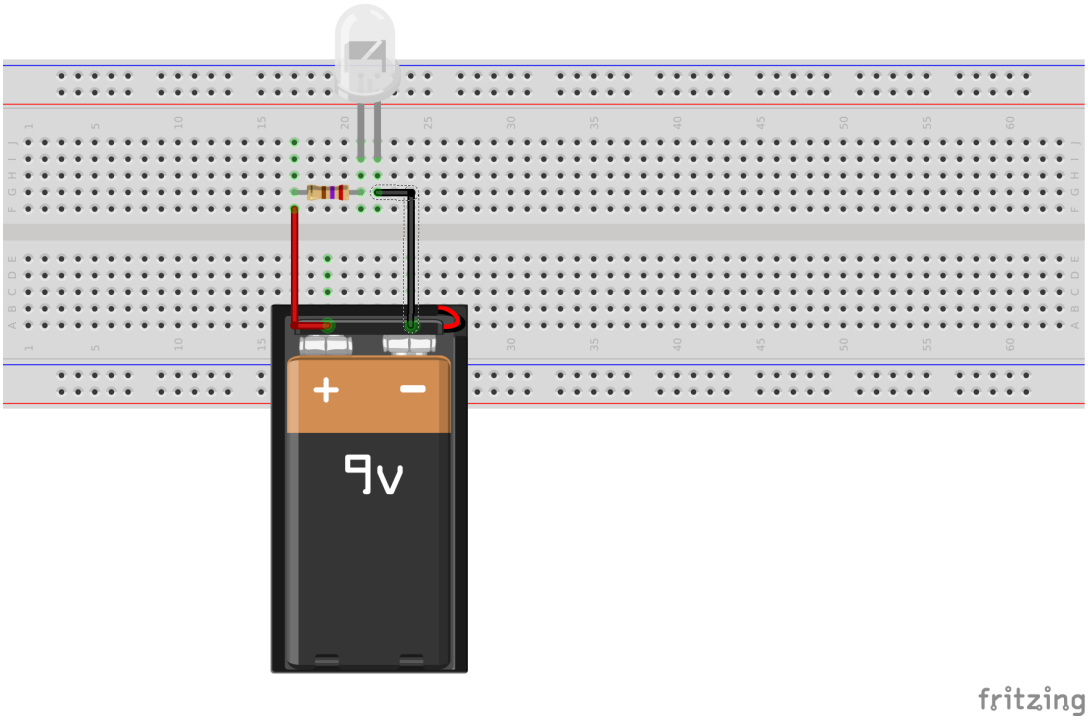

Figure 1: A simple example circuit demonstrating the use of a breadboard

- Micro-USB Power Cable

Hint: *Be sure to connect the peripherals before attaching the power to the Pi, as not doing so may result in the display not being detected properly.*

2.2 Getting Started with Raspbian

As you will have noticed, once the Pi is provided with power, it automatically starts to boot into the operating system on the SD card. Once booted you will be provided with a text based interface by default and be asked to log in. To so perform following steps:

1. Provide your username (the default user on the provided cards is "user1") and hit return
2. Provide your password (again the default password on the provided cards is "user1") and again hit return

Hint: *When typing your password, you will not see any text appearing on the screen. This is normal and is by design.*

At this point you will be provided with a Bourne Again Shell (BASH) "prompt" with some text and a flashing cursor at the end of the text. This should look similar to figure 2. This prompt shows the logged in user, the name of the raspberry pi and the current directory. Any commands entered will appear after the "$" and can be executed by pressing return.

```
user1@raspberrypi:/home/user1 $
```

Figure 2: An example of a BASH terminal prompt.

In addition to this text based environment, the Raspberry Pi provides a graphical user interface, that can be started by typing in: **startx** at the prompt and then pressing return. Try starting the GUI.

2.3 Installing Python

By default, Raspbian comes pre-installed with python 2. On this course, we are using the latest version of python (python 3) and this needs to be installed. To do so issue the following commands:

```
sudo apt-get install python3
```

Hint: *You may be asked for your password to continue. This is normal as you need administrative privileges to install software. You may have noticed that we used the term sudo at the beginning of the command, which is what indicates that your command should be run as an administrative user.*

The output from this command should indicate that the software was successfully installed.

2.4 Installing PyCharm

As you will have seen in the lecture, on this course we shall be using the PyCharm software for developing your python code. While this is not required, an integrated development environment (IDE) will provide some additional convenience over a basic text editor.

To install PyCharm you need to issue the following commands:

3 A First Exercise

Now that you have your Raspberry Pi prepared to write some python code, you will assemble a circuit consisting of an LED and a resistor on a breadboard. You will then write some python code to make the LED blink. This will allow you to check that your GPIO breakout board has been assembled correctly and that all of the software is configured correctly.

3.1 Preparing the Pi

To begin, we firstly need to install the software library that will be used to interact with the GPIO from within the python environment. To do this, you will need to open a terminal using the provided icon on the desktop. You will then need to enter the following command:

```
sudo apt-get install python3-rpi.gpio
```

Again, the output from this command should indicate that the software was successfully installed.

3.2 Building the Circuit

The next step is to build a circuit involving the GPIO breakout board, an LED and a resistor on the provided breadboards. This should be wired as shown in figure 3. Pay special attention to the LED, which need to be inserted in the breadboard the correct way round. The positive leg of the LED, can be identified as the longer of the two legs and should be connected on the same row as the resistor.

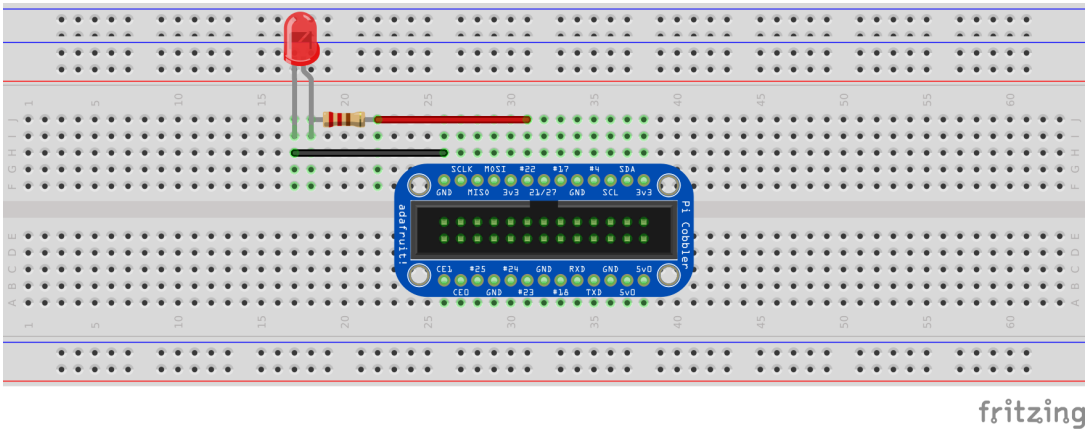

Figure 3: A circuit involving a single LED

3.3 Writing the Code

We now need to write some code to control the state of the output we have used. This will in turn allow current to flow to the LED and the LED to light. In PyCharm, start a new project and create a new file. Proceed by making the python file look similar to the following:

```
# Import time
import time
# Import the required module.
import RPi.GPIO as GPIO
# Disable Warnings
GPIO.setwarnings(False)
# Set the mode of numbering the pins.
GPIO.setmode(GPIO.BCM)
# GPIO BCM22 is the output.
GPIO.setup(22, GPIO.OUT)
# Initialise GPIO BCM22 to low (true) so the LED is off.
GPIO.output(22, False)
# Declare variable to hold state of LED
state = False
while 1:
    GPIO.output(2, state)
```

```
state = not(state)
time.sleep(0.5)
```

4 Temperature Measurement Circuit

By performing the last exercise, you should have made sure that your GPIO breakout board is working as expected and that the drivers for interacting with the GPIO were installed correctly. We will now proceed to explore a more advanced application of the Pi as an instrument to read the current temperature.

4.1 Building the Circuit

The Pi alone is not equipped with a temperature sensor and thus we will need to attach one. In this instance we will be using a DS18b20 sensor that communicates with the Pi using a one wire interface. In addition this sensor requires a resistor to be inserted between the data-wire (the yellow wire) and the power source (the red wire). To get started, assemble the circuit as shown in figure 4.

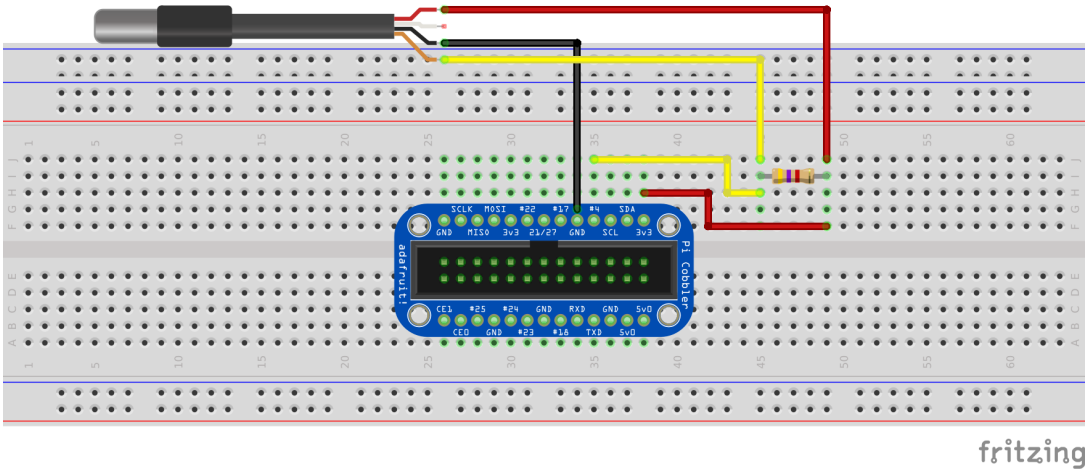

Figure 4: A circuit involving a temperature sensor

4.2 Preparing the Pi

While we have installed a python library that will allow us to control the GPIO ports from our code, we will need a more advanced module to communicate with the temperature sensor. This module is included with Raspbian but needs to be enabled. To do so enter the following commands:

```
sudo sh -c "echo dtoverlay=w1-gpio >> /boot/config.txt"
sudo sh -c "echo w1-gpio >> /etc/modules"
```

```
sudo sh -c "echo w1-therm >> /etc/modules"
sudo systectl reboot
```

The first three commands enable the one wire interface within the operating system. The last command reboots the Pi so that the changes will be applied. Once rebooted, log in and issue the **startx** command to return to the GUI. Now that you have done this you should check that the sensor is configured and working correctly. To do so you can enter the following commands in a terminal:

```
cd /sys/bus/w1/devices/
ls
# In the list of files and folders returned look
# for a result that looks like 28-xxxxxx.
cd file_from_last_command
cat w1_slave
```

The **cd** command changes your current working directory, the **ls** command allows you to see all of the files in the current directory and the **cat** command is used to read a file. When reading the file you should receive a result that looks similar to the following:

```
pi@raspberrypi:/sys/bus/w1/devices/28-0316457c94ff $ cat w1_slave
ad 01 4b 46 7f ff 0c 10 b3 : crc=b3 YES
ad 01 4b 46 7f ff 0c 10 b3 t=26812
```

The pairs of letters and digits are the data returned by the sensor. The Yes at the end of the first line indicates that the temperature was read successfully by the Pi and can be found on the next line. The number following the t= is equivalent the temperature measured x 1000 (in this case the temperature is 26.812).

You may also have noticed that to read the data from the sensor, we read a file. This is one of the specific properties of Linux, that all devices are treated as files, including serial ports, hard disk drives, etc.

4.3 Writing the Code

Now that the temperature sensor is attached and working, it should be possible to read the temperature from within python. An example of a python program that can be used to communicate with the temperature sensor has been provided and should be loaded inside the PyCharm IDE.

When starting the program, you should find that the temperature is printed out. As you heat or cool the temperature sensor, you should see this reflected in the values displayed by your program.

4.4 Further Exercise

While it may be useful in some instance to simply execute your code from within the IDE, at some stage you may wish to run the code separately or log the output from your program to disk. We will now look at how this can be achieved.

To get started, open the terminal that we used to install the packages previously. Using the change directory command that we used before, switch to the directory where you have stored your python code. To run you code you can then type the following:

```
pi@raspberrypi:/home/user1/development $ python3 name_of_file.py
```

You should now see the output from the program printed to the terminal. To end the execution of the program, you can press Ctrl+C. This will bring you back to the command line and give you a BASH prompt.

Lastly we will log the output of the program to file. To do so modify the command used to run the program to look like the following and press return to execute it:

```
pi@raspberrypi:/home/user1/development $ python3 name_of_file.py > output_file
```

Again end the execution of the program by pressing Ctrl+C. You should now be able to view the data that you have logged using the **cat** command:

```
pi@raspberrypi:/home/user1/development $ cat output_file
```

5 More Advanced Temperature Sensor Applications

During the last two exercises, you have learned how to control the state of the general purpose output pins and read the temperature from a one-wire temperature sensor. As an additional exercise, you should now try and combine the two circuits presented and the two programs that you have written. The result could be a program that measures temperature and turns on and off an output (you could use an LED) when the rises and falls against a set threshold. Bear in mind that you may not want the output to be quickly turning on and off if the sensor reading is changing frequently. How could you change your program to avoid this.

An example of such a program will be made available at the end of the lab session.

Mr. Adam Ziolkowski, Dr. Mark Fisher
Jan. 2017