Chapter 1

Introduction to RPi3 Model-B

This document includes a small introduction to how to access GPIOs in Raspberry Pi boards. In addition, we will use the document "**How_Tos.pdf**", which can be found on the Canvas website, under **Modules→Other Materials**.

1.1 General Purpose I/O (GPIO)

The Raspberry Pi 3 has many I/O ports and a total of 28 Digital, general purpose I/O lines available which are located on the header labeled "GPIO". All of the I/O lines can be programmed individually as inputs or outputs. There are many registers associated to the GPIO ports, which are aligned on 32-bit (4 byte) boundaries.

To use the ports as inputs or outputs, they need to be configured accordingly. The Function Select registers (GPFSEL) are used for that purpose. Modifying individual bits in the GPFSEL registers changes whether the corresponding I/O pins are inputs (default) or outputs.

When configured as inputs, the value of the GPIO pins can be read from the GPIO Pin Level (GPLEV) registers. When configured as outputs, the GPIO Pin Output Set (GPSET) registers are used to set the pins, and the GPIO Pin Output Clear (GPCLR) registers are used to clear the pins.

For more details, you can check chapter 6 on the "BCM2837_ARM-Peripherals" document. Be careful, that document has some typos and errors.

In order to use the I/O registers in our applications, we need to map these register's addresses into our program memory. In user space programs, this is done by using the mmap() function. This function requires a file descriptor, so we need to open up the special file dev/mem in our program first, using the open() command. Since we are mapping 32-bit registers into our program, we can use an *unsigned long ptr* for access to each port.

Doing the mapping above and accessing the registers directly can be a bit tricky at first. Fortunately, there is a nice library called **WiringPi** (`https://projects.drogon.net/raspberry-pi/wiringpi/functions/`) which provides an API with many functions that make interfacing with the GPIO ports a lot simpler. WiringPi also includes a command-line utility, gpio, which allows to program and setup the GPIO pins directly from a terminal or using shell scripts.

Turning On/Off the LEDs

If you are planning to turn of/off LEDs, you could make use of the following function available in **WiringPi** library.

```
int wiringPiSetup(void)
int wiringPiSetupGpio(void)
void pinMode(int Pin, int Value)
void digitalWrite(int Pin, int Value)
```

Notes: You need to include the wiringPi.h header file in your source code, and you need to include the **wiringPi** library when linking your program (**-l**). *Eg:*

```
gcc MyLED.c -o MyLED -lwiringPi
```

Push Buttons

In order to access push button, you might need the following functions functions along with the ones mentioned earlier:

```
void pullUpDnControl(int Pin, int pud_mode)
int digitalRead(int Pin)
```

1.2 General notes for Raspberry Pi 3 development

You may use BCM2837-ARM-Peripherals.pdf for most development. Even though it says BCM2837 it is almost same as BCM2835 document. The Pi3 is a BCM2837. Either of the documents still applies to the BCM2837, with the exception of ARM control block, which is almost the same as that in the BCM2836.

On the Internet, you may find "the Pi3 is like the Pi2 except for some minor details, use the Pi2 docs". Then, you may find "the Pi2 is like the Pi1 except for some minor details, use the Pi1 docs". However, be careful, **MEMORY LOCATIONS ARE DIFFERENT**, and that can be frustrating.

The GPIO base address is 0x3F200000
The Interrupt base address is 0x3F20B000

You may also check the GPIO-Pads-Control2.pdf, which explains features absent from Broadcom's documentation.

There may be some typos/errors on Broadcom's documentation.

The bcm2835 C library is also a useful reference (for example, it was reverse-engineered to figure out base addresses).

The WiringPi library (already available on the Lab's RPis) is useful for user space and terminal-based manipulation. There is a library for the ADC used by the auxiliary board.

Chapter 2

Auxiliary Board

You will have access to an auxiliary board, which can be connected to the RPi3. This auxiliary board contains five push buttons, three LEDs, and a speaker. The push buttons should be connected to GPIO ports configured as inputs, and the LEDs and speaker should be connected to GPIO ports configured as outputs. Figure-2.1 shows an image of the auxiliary board.

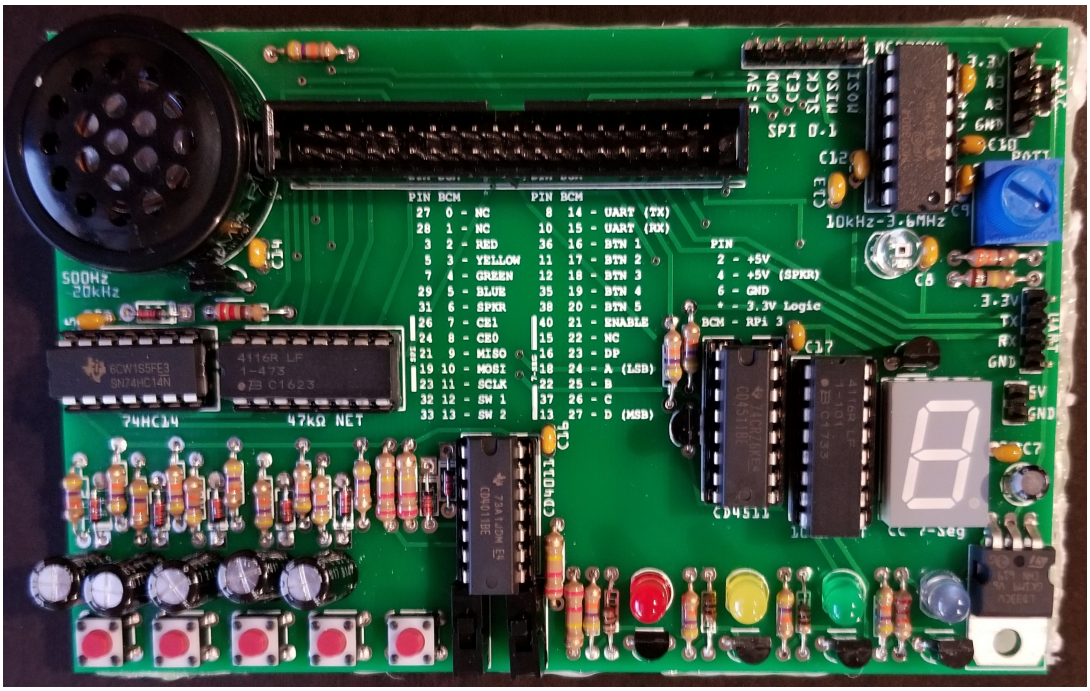

Figure 2.1: Auxiliary Board

WiringPi uses its own GPIO numbering scheme. You need to be careful when connecting your boards, and when using the wiringPi API. Figure-2.2a shows the RPi3 GPIO pinout, and Figure-2.2b shows the wiringPi GPIO pinout.

(a) RPi3 pinout

Raspberry Pi 3 Model B (J8 Header)

GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1	2	5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	4	5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5	6	Ground	
7	GPIO 7 GPCLK0	7	8	GPIO 15 TxD (UART)	15
	Ground	9	10	GPIO 16 RxD (UART)	16
0	GPIO 0	11	12	GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	14	Ground	
3	GPIO 3	15	16	GPIO 4	4
	3.3 VDC Power	17	18	GPIO 5	5
12	GPIO 12 MOSI (SPI)	19	20	Ground	
13	GPIO 13 MISO (SPI)	21	22	GPIO 6	6
14	GPIO 14 SCLK (SPI)	23	24	GPIO 10 CE0 (SPI)	10
	Ground	25	26	GPIO 11 CE1 (SPI)	11
30	SDA0 (I2C ID EEPROM)	27	28	SCL0 (I2C ID EEPROM)	31
21	GPIO 21 GPCLK1	29	30	Ground	
22	GPIO 22 GPCLK2	31	32	GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	34	Ground	
24	GPIO 24 PCM_FS/PWM1	35	36	GPIO 27	27
25	GPIO 25	37	38	GPIO 28 PCM_DIN	28
	Ground	39	40	GPIO 29 PCM_DOUT	29

Attention! The GIPO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

http://www.pi4j.com

(b) WiringPi pinout from http://pi4j.com/pins/model-3b-rev1.html

Figure 2.2: Raspberry Pi Pinout