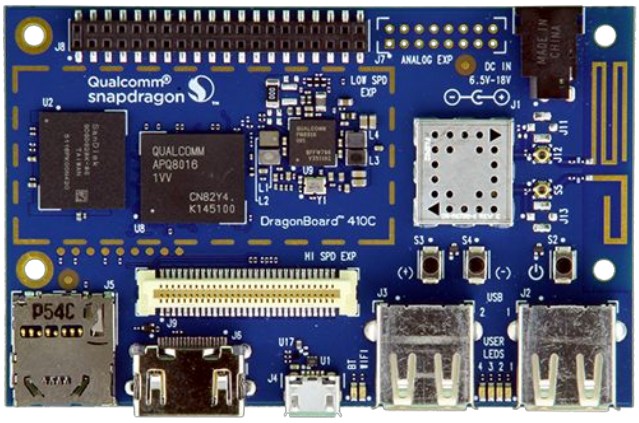


DragonBoard™ 410c

Module 2

GPIO Programming

Table of Contents

GPIO Programming

In order for the DragonBoard™ 410c to interact with the world there has to be an interface between them. For the purpose of this project, the GPIO interface will serve as a way to sense and interact with the environment. This document will talk about General Purpose Input/Output pins and why they are important to this project. It will try to define them as well as provide other resources that could help further explain their purpose. In taking a look at the low speed expansion header on the DragonBoard™ 410c we will locate and explain all other GPIO interfaces. Since only the 12 GPIO will be necessary for this course, most of this document's focus will be on them. Once a greater understanding of the GPIO is achieved we will then access them via command prompt be it through a PC host or on board OS such as Ubuntu. Finally this document will show you how to make your first program/application capable of controlling a GPIO.

Name: GPIO Programming
Project Description: **Students will be required to watch all videos and do some research on a variety of GPIO interfaces. Student will then access a GPIO through a terminal/command prompt and perform some simple commands. Once familiar with the functionality of the GPIO, students will create their first application/program to control one or more GPIO pins and make them perform more a complex sequence.**

Code Access:
The code for this module is found at the following link:
https://github.com/IOT-410c/IOT-DB410c-Course-3/tree/master/Modules/Module_2_GPIO_Programming

To obtain this code, issue the following command:
git clone https://github.com/IOT-410c/IOT-DB410c-Course-3

Then inside of Android Studios, import the project found at
IOT-DB410c-Course-3/Modules/**Module_2_GPIO_Programming**

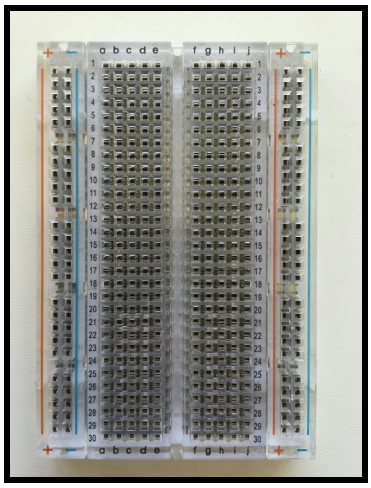
Note:

- **Ensure the DragonBoard™ 410c is connected to a display**
- **Ensure the correct power source is connected to the device (DC 12V, 2A)**
    - **WARNING**: Exceeding the recommended power could damage the device
- Ensure the device has USB debugging enabled (Android users only)
- Ensure the DragonBoard™ 410c is connected to a computer

*We encourage all who read this document to visit the websites that are provided. Please use these resources to gain a deeper understanding on the subjects at hand.

Parts List:

a. 400 point breadboard (Optional)
b. Miscellaneous wire (Optional)
c. LED (Optional)



a.

b.

c.

---

1 - General Purpose Input/Output (GPIO)

Understanding what general purpose input/output pins are is what we would consider to be one of the foundational building blocks for this course.
This section will outline the basics of general purpose input/output pins and relate them to the specific projects found in this course.

1.1 - GPIO Defined

General purpose Input/Output pins or GPIO are pins that go generally unused by default and are said to have no defined special purpose. This means **the user maintains decisive control over the GPIO pins and their actions**. That being said, these GPIO are capable of performing a variety of user driven actions, most of which will be necessary for this course. Below is a list of potential capabilities of the GPIO pins as seen on
https://en.wikipedia.org/wiki/General-purpose_input/output

GPIO capabilities may include:

- GPIO pins can be configured to be **input or output**
- GPIO pins can be **enabled/disabled**
- Input values are readable **(typically high=1, low=0)**
- Output values are **writable/readable**
- Input values can often be used as **IRQs (Interrupt)**, typically for wakeup events

Here it is important to note that the GPIO pins are configurable, and can be set as an input or output. With that, we see **values can be written onto, or read from these interfaces (GPIO), typically as discrete values of 0 and 1 (High or Low)**. Being able to read and write values to these pins allows simple and quick communication with peripheral devices. These devices in turn help the DragonBoard™ 410c to interpret and communicate with the environment or other devices. All microcontrollers are not the same, and will usually differ in many ways. **The DragonBoard™ 410c has 12 GPIO pins, one of which is multi-purpose.**

1.2 - GPIO, Project Specific Uses

GPIO interfaces will be utilized to control the many circuits and components throughout this course and the others. Below is a list ways we will use the GPIO interfaces for this course:

- One GPIO will be used to toggle an LED (on/off)
- Four GPIO will be used to control a stepper motor
- Three GPIO will be used to control the LED matrix (1-8 blocks)
- One GPIO will be used to toggle a PIR (Passive Infrared) sensor
- Seven GPIO will be used for the IR (Infrared) remote control circuit.
    - One for the IR receiver
    - One for the PIR sensor circuit
    - Four for the stepper motor
    - One for the indicator light
- Five GPIO for the ultrasonic sensor (Three will be used for proximity LED's)
- Ten GPIO for the Bluetooth remote
    - One for the PIR sensor circuit
    - Four for the stepper motor
    - Five for the Ultrasonic (Three will be used for proximity LED's)

Please note:

1) At any given time there will only be access to 11 GPIO pins. These projects will be built and run separately leaving plenty of GPIO for each project.
2) In most cases, while using a GPIO as an output, it will be required to go through the amplifier built in Module 3.

For this project, we will be using the pins to:

- turn on and off a LED
- controlling a LED matrix
- interface with various sensors, such as the ultrasonic sensor, passive infrared (PIR) sensor, etc.
- stepper motor control
- Bluetooth controlled outputs

1.3 - GPIO, Overall Uses

Some other things you can use the GPIOs for (person projects):

- Robotics applications
    - Motor control: driving, walking, running, propellers (boats)
- create feedback loops
    - start, stops, interrupts
- most forms of I/O (as long as it's digital)

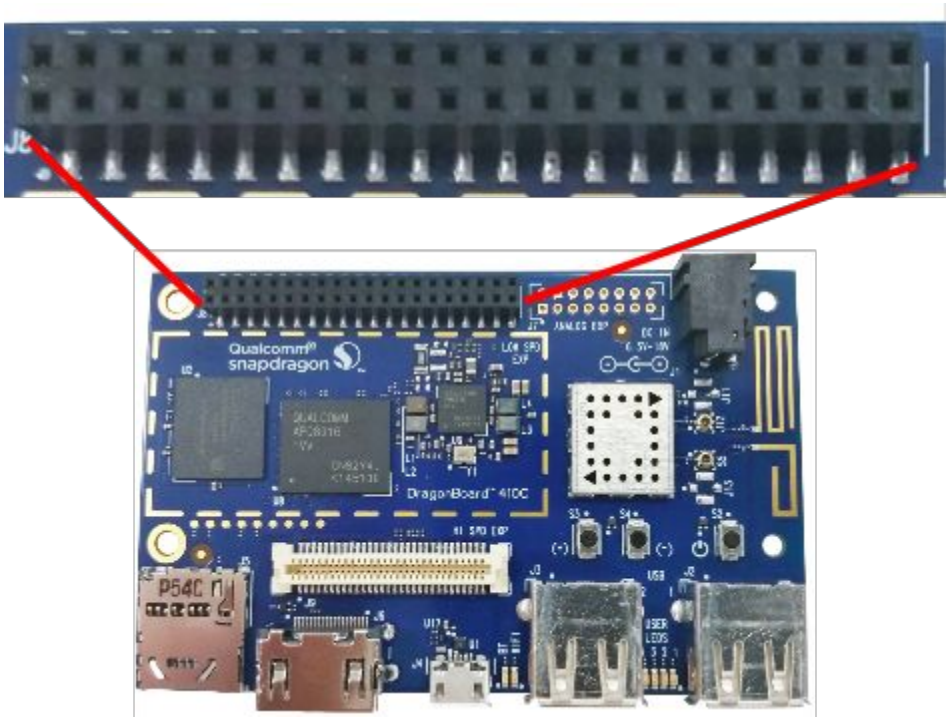
Although this is a short list, many other projects you can take: challenge yourself or find something cool online!

2 - The Expansion Header

This section will teach you what the expansion header is and how to access it. It will also show you the pin layout for both the Android OS and Ubuntu.

2.1 - Pin Layout

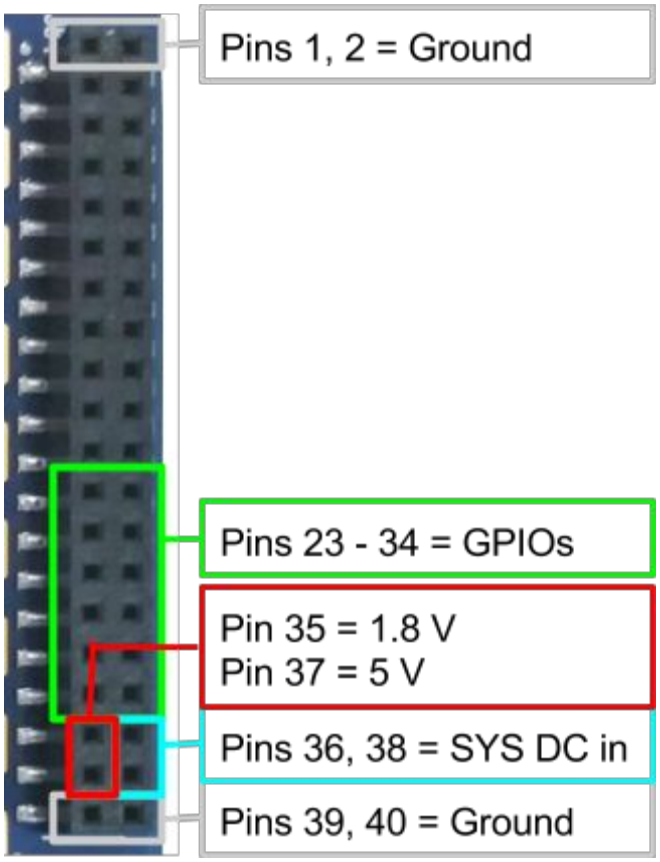
40 pin low speed expansion header can be found on the board here (black protrusion, next to the analog expansion header holes):



The expansion header is the home for the main interface we will be using: general purpose input/output (GPIOs). This header also houses an MPP, the SPI, I2C, UART, and PCM pins. This document will go over all of these interfaces in the next lesson.

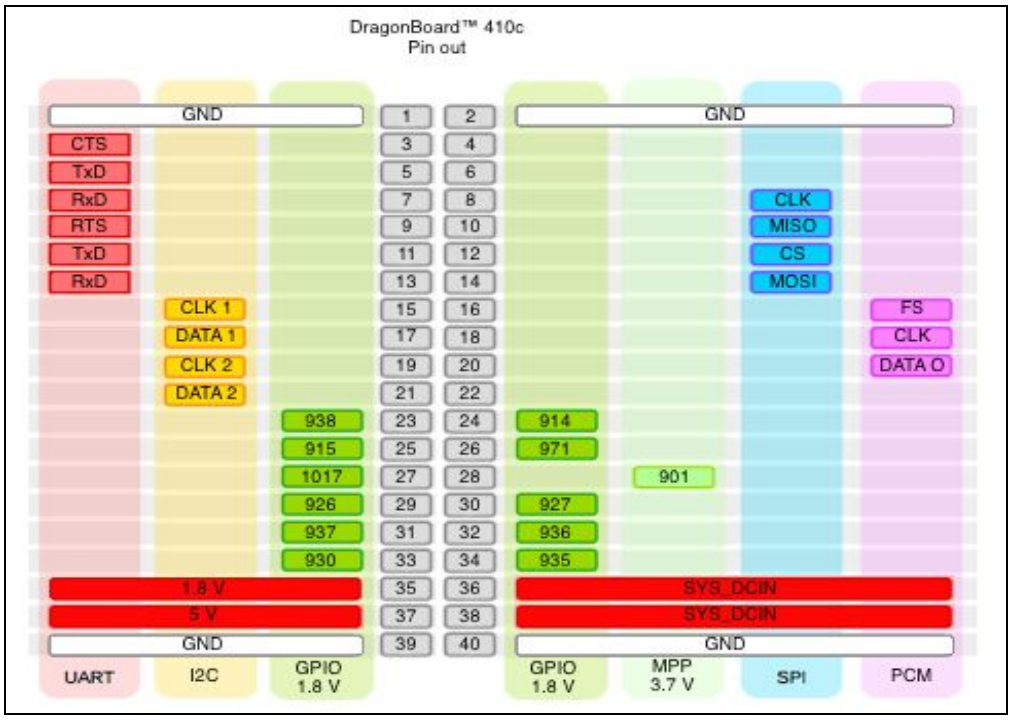
The diagram on the right shows the physical locations of the important pins we will be using later on. (**Note**: Pin 1 is on the top left, far away from analog header, and pin 40 is on the bottom, close to the analog header). Here is a list with the associated color on the diagram:

- Ground: corners of expansion header; can use any ground
    - pins 1, 2, 39, 40 (gray)
- 1.8 V power supply
    - pin 35 (red)
- 5 V power supply
    - pin 37 (red)
- System DC in: supplies the same amount of power as your DC adapter supplies to the board (6.5 V to 18 V) - useful when 1.8 V or 5 V isn't enough
    - pins 36, 38 (light blue)
- GPIO pins
    - pins 23-43 (gree)
    - Note that pin 28 is an exception, it is an MPP instead of a GPIO

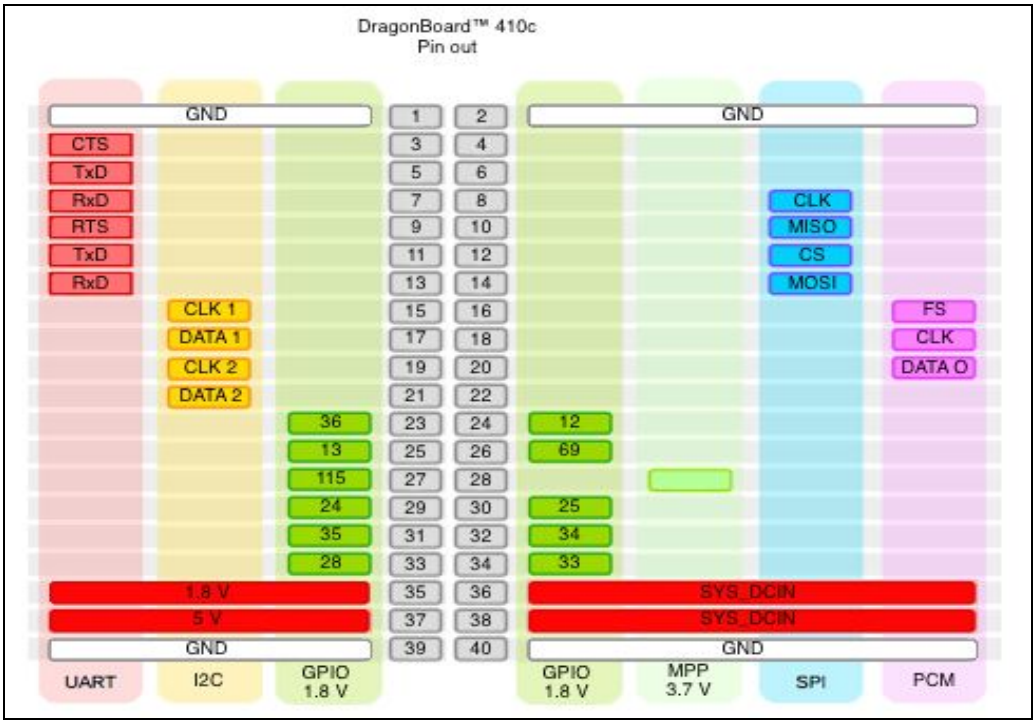


Now that you know where the pins are physically located, how do we work with them on the DragonBoard™ 410c? This answer will depend on which OS your board is running:

If your board is running the Android OS, you will need to use these numbers to access the pins on the board:



And here is the diagram for the Ubuntu OS (Android # - 902 offset):



2.2 - Pin Access

The pins can be accessed in two ways: single pin access (using individual wires) or through a ribbon cable breakout board. A breakout board will make accessing the pins easier. Note that the connector headers on the Dragonboard™ 410c are 2mm headers and not the usual 2.54 mm headers.

3 - Other GPIO Interfaces

This section will teach how to use the other interfaces and components on the Dragonboard such as the MPP, I2C, UART, SPI, and PCM.

3.1 - MPP vs. GPIO

The multi-purpose pins (MPPs) are work very similar to GPIO pins but can be used for more functionalities. They can be used as analog Input/Output. This means that instead of only being able to detect or output a high or low signal, the MPP can detect a wider range of signals as well as output more information. However, the pitfalls of an MPP are that it is slower and occupies more space on the memory. (**slides)**

3.2 - I2C

The Inter-integrated Circuit (I2C) is essentially a serial computer bus (a bus is something that communicates/transfers data between components) which allows lower speed peripheral ICs to be connected with processors and microcontrollers. They permit multiple synchronized master-slave connections to be formed. It is used in short distance communications within a single device. Although it is slower, it can be used for many devices. Requires two signal wires to transfer information.
https://en.wikipedia.org/wiki/I%C2%B2C

3.3 - SPI

The serial peripheral interface (SPI) is a bus that is used for fast, short distance communication between two systems in a master-slave fashion. The master device will contain the original signals which can be processed in the slave. The main difference between SPI and I2C is that SPI is a 4 wire interface while I2C is a two wire interface. SPI is used for guaranteed data transmission while I2C is easier to link multiple devices for it has a built in addressing scheme.
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

3.4 - UART

A UART is a microchip that controls the interface between a computer and its attached serial devices. It translates data between parallel and serial. It can talk to and exchange information between modems and other serial devices. It is asynchronous (requiring a form of computer control timing protocol). The main difference between UART and SPI is in their protocols and speed. SPI typically operates at up to 50 Mbps while the UART operates between 0.5 Kbps and 1 Mdps. Therefore, SPI is excellent for high-bandwidth applications, such as audio and video. UARTs is better for sensor type applications.
https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
https://gigaom.com/2008/08/12/why-we-need-fat-pipes-the-top-5-bandwidth-hungry-apps/

3.5 - PCM

Pulse-code modulation (PCM) is used to sample analog signals and convert it to a digital signal. The analog signal is, in effect, compressed into a digital signal (not perfect analog representation but close) which means that the data is easier to work with (storing, reading, less noise). **https://en.wikipedia.org/wiki/Pulse-code_modulation**

4- Access GPIO through Terminal

This lesson will teach you how to access (low/high) the GPIOs through the terminal for both Android and Ubuntu OS's.

4.1 - GPIO Access with Android (SDK Recap)

Click here to skip to the Ubuntu GPIO access.

- To access the GPIO through Android, first make sure the board is connected
- Travel to your Android SDK folder
- Find where the adb.exe is located, usually inside the platform-tools folder.
- start adb through the terminal or type **./adb** if being run through computer terminal
- use **./adb devices** or **adb devices** to check if your device is being detected
- **./adb shell** or **adb shell** to enter your device
- execute "**su**" to enter superuser mode (be careful now)
- travel to the GPIO folder: **cd /sys/class/GPIO**
- issue ls command to see all the gpio pins you have access to at the moment
- To actually use the GPIO use the following commands
    - **echo [**Android Pin **#] > export**
        - issue **ls** command and you should see that pin come up titled "gpio#"
    - **cd gpio#**
        - another **ls** command will show you everything you have access to for this pin
    - **cat direction**
        - this will return if this pin is in output or input mode
    - **echo [**out or in**] direction**
        - out sets the GPIO to output mode
        - in sets the GPIO to input mode
    - **cat value**
        - this will return 0 if the GPIO is off, 1 if it is on
    - **echo** [0 or 1] **> value**
        - 0 to turn it off
        - 1 to turn it on

4.2 - GPIO Access with Ubuntu

To access the GPIOs on Ubuntu, open the terminal (start menu > other > LXTerminal; right click this to add to desktop if you desire) on the Dragonboard and travel to the GPIO folder through the following command: **cd /sys/class/GPIO**
Next, use **sudo su** to give yourself the ability to modify the GPIOs (if you get an access denied for a GPIO, it means you're trying to modify a GPIO that the board using for itself)
To actually use the GPIO use the following commands:

- **echo [**Ubuntu GPIO **#] > export**
    - this will give us the ability to work with a certain GPIO pin
- **cat direction**
    - this returns if the pin is in input or output mode
- **echo [**out or in**] direction**
    - out sets the GPIO to output mode
    - in sets the GPIO to input mode
- **cat value**
    - this returns 0 if the pin is off, 1 if the pin is on
- **echo [**0 or 1**] > value**
    - 0 turns the GPIO off
    - 1 turns the GPIO on

5a - Access GPIO through Programs (Android)

This section of the procedure document will guide the user as to how they can write their own scripts or programs to access and work with the GPIOs on the board using the Android OS.

5a.1 - Modifying Boot Script

In order to control the GPIOs through Android applications, we will need to modify a boot script. Once the boots script has been modified, every time we boot into Android, all usable GPIOs will be exported and granted the appropriate permissions for use within Android applications. Please follow the steps in the video, commands that are used can be seen below:

$ adb root

$ adb remount

$ adb shell

$ mount -o rw,remount /system

exit out of adb shell
(shell) $ exit

adb pull /etc/init.qcom.post_boot.sh

(add to the post_boot.sh)
```
set -A pins 938 915 1017 926 937 930 914 971 901 936 935
for i in 0 1 2 3 4 5 6 7 8 9 10
do
  echo ${pins[i]} > /sys/class/gpio/export;
  chmod 777 /sys/class/gpio/gpio${pins[i]};
  chmod 777 /sys/class/gpio/gpio${pins[i]}/value;
  chmod 777 /sys/class/gpio/gpio${pins[i]}/direction;
done
```

adb push init.qcom.post_boot.sh /etc/init.qcom.post_boot.sh

adb reboot

adb root

adb shell

cd /sys/class/gpio (to check if they're there)

5a.2 - GPIO Library

This library can be pulled using the appropriate Git link at the beginning of this document.

5a.3 - Make Your Application (Gradle)

Please review the video to gain insight on how to use grade for importing our GPIO Library

5a.4 - Make Your Application (Manually)

Please review the video to gain insight on how to manually import our GPIO Library

5b - Access GPIO through Programs (Ubuntu)

This section of the procedure document will guide the user as to how they can write their own scripts or programs to access and work with the GPIOs on the board using the Ubuntu OS.

5b.1 - Python and Other Languages

This sub-lesson will teach the user how to install Python along with a nice IDE known as IDLE3 so you can have an easier time writing your code in Python.

We chose Python over other languages because it's a great language to get started programming with. Python allows the user to get a small exposure to the logic and nuances of programming without having to worry too much about syntax. Hopefully, this way, any new programmers won't be overwhelmed with trying to learn a language (Java, C/C++, etc).

To begin:

- launch the terminal on your DragonBoard™ 410c (make sure it's connected to the internet)
- **sudo apt-get update**
    - this updates your files
- **sudo apt-get upgrade**
    - this upgrades your files with the new updates
- finally, **sudo apt-get install idle3**
    - this will get the newest version of the idle3 text editor

    - note if you don't like IDLE3, feel free to look up the Ubuntu package manager and install your favorite text editor

To begin using the IDLE3 ide, launch it and a new window with the Python command prompt should come up. To write scripts or text files, go to the top and click File > New File. This will bring up a new text document for you to write, save, and run.

5b.2 - GPIO Library

So now that you have your IDE, Python, and board all ready to go, we're almost ready to start building programs. But before you even build a program, you are going to need a GPIO library to access and work with the GPIOs through your programs. This is because as a program/script, it doesn't know how to work with the GPIOs on your board. Therefore, we have provided a file that makes it possible for you to interact with the GPIOs.

Basically, the GPIOLibrary.py file makes it easier to access the GPIO pins without having to worry about fixing everything in the terminal beforehand. The script handles all the setup for you and:

- you may simply use the getPin(#) function to get access to a GPIO for your program.
    - the # to use would be the pin # on the expansion header, the script will return the correct pin # with the corresponding Ubuntu pin #
- has getters and setters for you to easily change the states of the pins or get the status of the pins in your code (easy looping or condition checks, etc.)

5b.3 - Build Your Program

This section will teach you how to build a program to turn on an LED.

- for the circuit, you need to hook up an LED with a resistor to an amplifier
- connect pin 34 (gpio33) to the amplifier
- **Note:** We will show you how to build an amplifier in the next module

- with the circuit all setup, let's start programming
- Launch IDLE3 (or your favorite text editor)
- open a new text file (call it LED.py)
- save the file in the same location/folder as your GPIOLibrary.py
- import the library by adding this line at the top:
    - **from GPIOLibrary import GPIOProcessor**
    - **import time** (this is for using Python's time functions for delays or timing)

- construct the GPIOLibrary using
    - **GP = GPIOProcessor**
- in a **try-finally** block add:
    - **Pin34 = GP.getPin34();**
        - get a reference to pin34
    - **Pin34.out()**
        - set it to output mode
    - write a for loop to make it blink 10 times; here is the entire code:

```
# LED.py
# this is a comment
# they don't affect the code/program
# just for the programmer(s) to see

from GPIOLibrary import GPIOProcessor  # import from GPIOLibrary

GP = GPIOProcessor()  # create the GPIOProcessor for us to use

try:

    Pin34 = GP.getPin34()  # get Pin34
    Pin34.out()            # set Pin34 to output mode

    for i in range(0,10):  # for loop to make the LED blink 10 times
        Pin34.high()       # set 1 for pin34 (on/high)
        time.sleep(0.5)    # sleep for 0.5s
        Pin34.low()        # set 0 for pin34 (off/low)
        time.sleep(0.5)    # sleep for 0.5s before repeating

finally:
    GP.cleanup()  # properly exit the program
```

Save the file, and locate it inside your terminal. After you've found the file, execute it using this command: **sudo python LED.py**
Don't forget the sudo because you need root access to work with the GPIOs.

With that, you have your first working program!

References