Final Project CSC155 (Robi)

# Project Objectives

The purpose of this project is to have students write hardware drivers that will allow Robi to move and avoid object. This file will contain all of the specifications that will be required in order to complete this lab. Robi should be able to move, as demonstrated in the video provided in Blackboard for this course.

# Project Instructions

On Blackboard, students will be given Python files that they must install on their Robi robot. These Python files will be used for the following with this project.

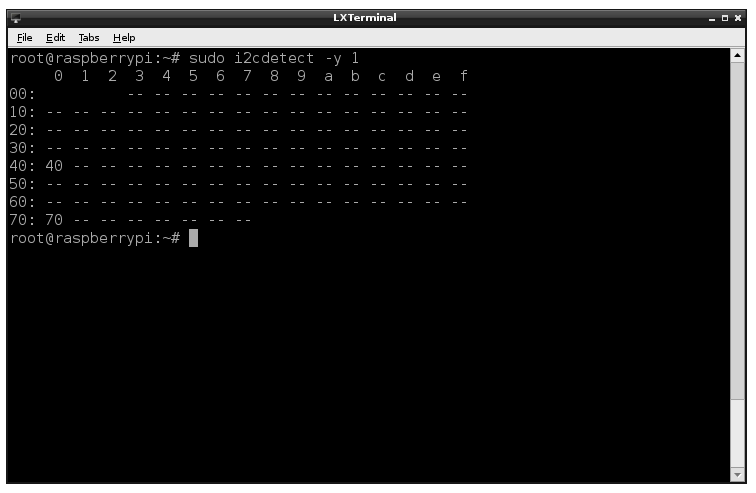
1. The files will be used to test Robi (they work). Running the Python files will ensure that the i2c bus is enabled on Robi and that the wiring for the DC motors is correct. The objective of using the files, is to show the students how Robi should work, when testing their C libraries.
2. The files are also readable and becomes a Guide for each student, as they write their C library files.

# Steps Completing this Project (overview)

1. Download from Blackboard the python files for Robi
2. Transfer the Python files to the pi user home directory on Robi
3. Check to see if i2c is working correctly on Robi before continuing. You do this by following the instructions located at

<https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c>

1. After completing step 3 you should have the image like the one on the next page (remember there is two hats, so you will have more numbers)



1. You then Will Run the Python files to test your Hardware and Make sure that everything is connect properly.
2. You will then start writing your C libraries in the order that is given to you. This will be difficult, since you are writing for hardware communications, and you will not have meaningful output to debug. Use the Python Drivers as a Guide.
3. Once you have completed writing your drivers and have Robi Moving, you will then be able to write the code for his sound sensor.

# Running Python Files as a test

1. Once you have completed the i2c verification for Robi, turn the hats on by using the power switch (some switches are on the front of Robi and some are on the battery casing).
2. You should download the Python files that are needed for this testing. You can find these files in Lesson 5 under Topic Material. The file is called hat.tar
3. Once you have moved hat.tar to Robi, extract the files by using the tar command.
4. Now navigate into the folder that contains the Python Code for Robi. This will be located within the Adafruit folder and inside the examples folder.
5. Make sure Robi is suspended off the table (so it does not drive off the table) and run the python code RobotTest.py

sudo python RobotTest.py

1. Robi should move at this point and indicate that there are no issues for writing your drivers with the hardware.

# Writing your C files for the drivers

The instructor will be providing to you’re a beginning to your drivers. The following files will be located in Blackboard and students should down load them to Robi in a separate folder. The following is a list of the files that should be downloaded.

1. Download the C library templates from the Activities folder in Lesson 5. The file is called examples.tar
2. Once this file has been extracted you will see the following files
   1. PWM.h
   2. Robot.h
   3. I2C.h
   4. I2C.c
   5. Sensor.h
   6. MotorHat.h
3. If you notice, this is a lot of .h files with only one implementation file. This is by design for the project. I have given you a start into what files may be needed. Each of the .h files, will be needing an implementation file and you will be needing to add more functions to each and adding additional files. The I2C.h and I2C.c files have been written for your and do NOT need to be edited or changed. This file contains three really important functions.
   1. I2cInit() – this function is needed to initialize the i2c bus and MUST be called before reading and writing to the bus. There are two parameters. The only parameter that is needed is address, buss is not used and can be set to the default number provided in the Python file. The address that will be provided will be the address of the Motor Hat. The address for the motro hat is 0x60 and should be passed to this function before writing to the bus.
   2. Write8() – this is a function that will write one byte of information to a specific register within the hat. REMEMBER both the reg and the value is ONE BYTE and you should be using the unsigned char as a parameter type for both. This function needs to be completed by you and you need to follow the function provided below.

void write8(unsigned char reg, unsigned char value){

char buff[2];

buff[0] = reg;

buff[1] = value;

uint8\_t result;

result = bcm2835\_i2c\_write(buff,2);

}

A note on the functions that are used within this code. All information concerning the bcm2835\_i2c\_write() function can be found in the online documentation. Please read this documentation carefully. It will provide information on the results that are returned from using this function. Students are required to add to this function error checking, based on the results returned.

* 1. ReadU8() – This function is to be completed by you. This function has one parameter and this is the register that you will be reading from. The best solution to this, is to again use a 2 byte buffer, but only read from the first byte. The function you will be using is bcm2835\_i2c\_read(). The following are notes on the use of this function.
     1. The only values returned are going to be the error codes. Not the value stored in the register. Each student should be handling all of these errors.
     2. The first byte of the character array will be the register and the length is going to be 1 (even though buff is 2)
     3. The first byte of buff will contain the value from the register. So the value of buff[0] should be returned from this function.

1. You can test your functions with the following values
   1. Register 0x01
   2. Value 0x04
2. To test the i2c.c file you must call the following functions in this order. You should be providing a main.c file that will be using this library and print the values returned when testing. Again, when running your code, you will need to make sure the hats are on, and that you use sudo.
   1. i2cIntit()
   2. write8()
   3. readU8()