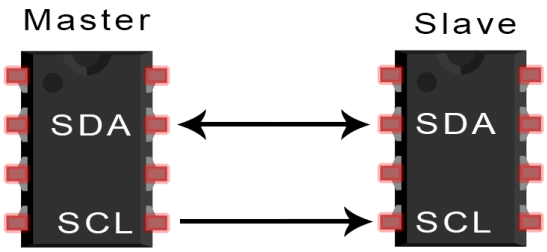
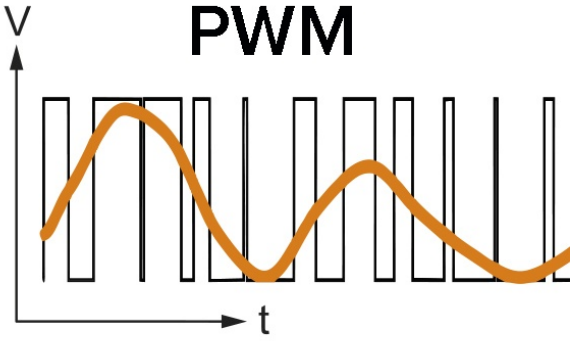
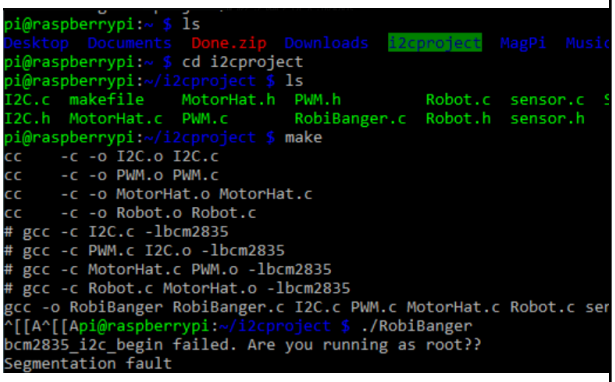
Robi Final Project

Description: In this project, I created a program that makes my Robi move forwards until it gets too close to the wall. When it does, it spins to scan directions and uses the sensors to determine if it should move left or right. It will then proceed forward in that direction, in my code it does this for an interval. I really enjoyed this project because it taught me a lot of useful information about integrating code to make my robot accomplish a given task.

Table Of Contents

Problems Encountered	Problem Solutions
	**i2c functions** We were able to successfully use the reference functions to complete the functions needed to write, read and initialize i2c so the channels could be established.
	**Pwm functions** We had to convert the milliseconds to full seconds and use bcm2835_delay() and sleep() to successfully convert the proper delays. Also the floor function was giving us trouble when compiling which we fixed by using the math library. We had to fix the parameters for all of these functions and remove .self along with python code in other files.
	**Error for not running as root** Solved by running as the root using sudo command.

Python casting

```
# Converting the string into an in
string_number = int(string)
```

C casting

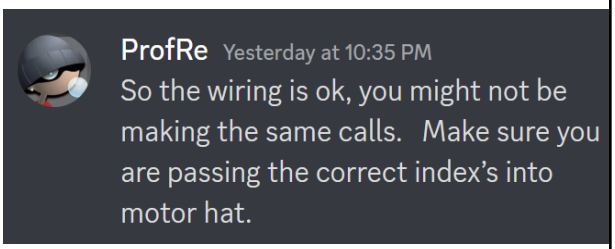
```
mean = (double) sum / count;
```

Casting Translation

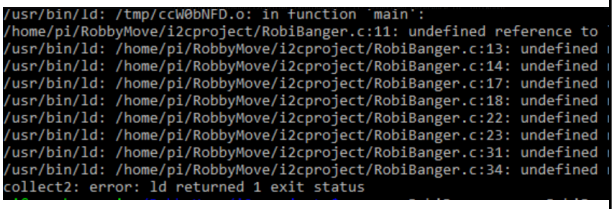
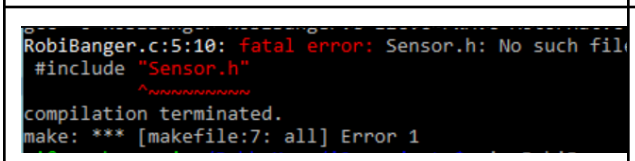
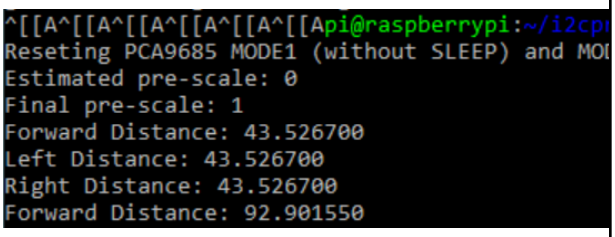
Had to fix the casting because it was causing an obvious error from not being the same in both languages.

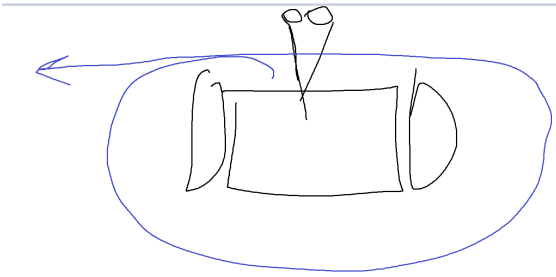
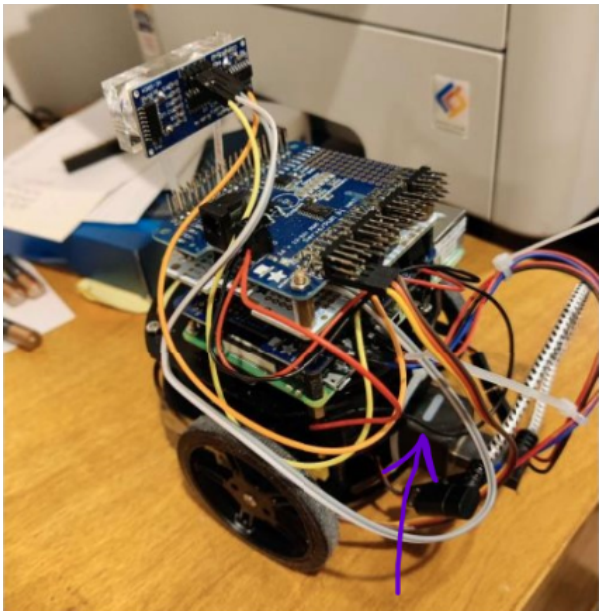
Translating

Thankfully converting most of the code from Python to C was not as troublesome as imagined besides some typos, missing ;'s, and this gave us little other issues.



It was time to start looking over the Python files once we had confirmed that our Robi was operational. The python files served as a great resource for figuring out what to do next. Despite the overwhelming number of files, Professor Re helped us and directed us

	in the appropriate route. What we discovered: - PWM utilizes I2C. - MotorHat uses PWM to operate. - Robot uses a MotorHat to move. - RobotTest utilizes Robot.
	**Compilation errors in make file** Solved by using -lm and compiling it all on one line.
	**#include errors for not having the correct file header** Fixed by correcting the header and sftping the sensor files.
	**Distances were the same when trying to make Robi choose a direction** So the program was not running correctly and he was not moving forwards or choosing a direction. Made an extra case for this which fixed it.

`` ^[[Api@raspberrypi:~/RobbyMove/i2cproject $ sudo ./RobiBanger Reseting PCA9685 MODE1 (without SLEEP) and MODE2Setting PWM f Estimated pre-scale: 0 Final pre-scale: 1 Waiting for sensor to settle Set TRIG to true Sleep Set TRIG to false First while loop pulse start: 604909.000000 Second while loop pulse end:608793.000000 pulse duration:0.003884 Distance:66.61 cm Current distance: 66.610600 Segmentation fault ```	**Segmentation fault** Not sure how I fixed it this just went away after I repasted my code back in from my VSCODE.
	**Robi over rotating** **Left and Right Rotate were also flipped** Fixed by making him do a complete rotation to face given directions a bit more accurately Left and Rotate were also flipped, so we just used the inverse. ←– Solution idea image
	**Did not have a visual battery on my robi like Justins (image)** Made it extremely difficult to know when my Robi was at full charge or low even if I just had him on the charger. So I would end up getting disconnected from my SSH when trying to change my code a lot.

MotorHat.c ● RobiBanger.c C:\...\Rar$DIa0.550 ● C

C: > Users > ninja > AppData > Local > Temp > Rar$DIa0.167 >

```
#include "MotorHat.h"

extern void initMotors(int num){

    if (num = 0){
        motors[0].pwm = 8;
        motors[0].in2 = 9;
        motors[0].in1 = 10;
    }
    else if (num == 1){
        motors[1].pwm = 13;
        motors[1].in2 = 12;
        motors[1].in1 = 11;
    }
    else{
        printf("MotorHat Motor must be bet
    }
}
```

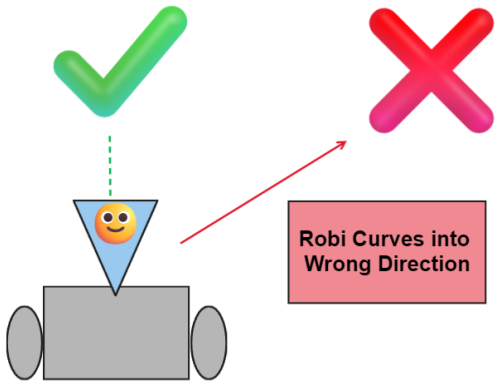
Cursed typo that caused a significant amount of code revision and code validation. Also caused only one of the wheels to not spin at all because the values were not changed.

Changed “num =” to

“num ==”

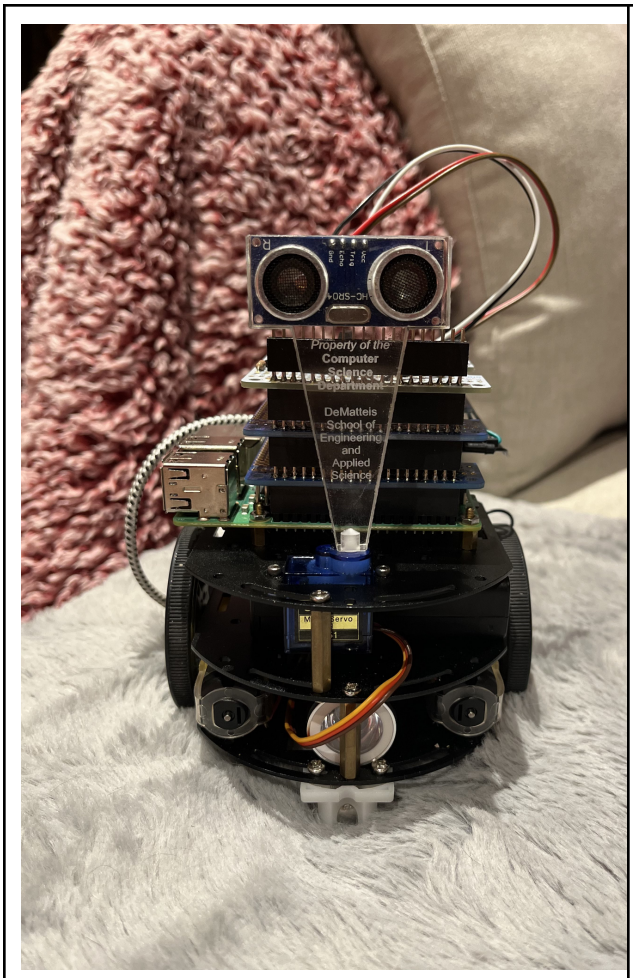
This boolean operator was used incorrectly instead of as a comparison making the code not work.

Found through adding print statements to print values.



Motors not accurate so the robot would curve slightly when moving

Did not end up fixing since the motors are not 100% accurate.

	**Needed to configure Sensors for the Extra Credit** My testing took me a lot longer than expected as the motors still would not move with perfect accuracy. Thus I had to keep changing the motor speeds and time lengths. Added and configured the sensors into our test code and used it within the makefile after downloading the C Sensor files from blackboard. Me and Justin decided to utilize the same test file naming convention as “RobiBanger” (to end the semester with a bang!).

Research Component

- Researched how to get my Robi working on my home WIFI so I could connect easier using SSH so I would not have to connect using HDMI every time which was a lot more difficult to use Robi when doing it this way.
- Looked up makefile errors, how makefiles work with gcc, and how to fix them.
- Examined segmentation faults.
- Researched C syntax for the library of atexit() function.

Works Cited

"C - Where to Add the -Lm Flag within a Makefile?" *Stack Overflow*, stackoverflow.com/questions/23243235/where-to-add-the-lm-flag-within-a-makefile. Accessed 18 Dec. 2022.

"C Library Function - Atexit()." *Www.tutorialspoint.com*, www.tutorialspoint.com/c_standard_library/c_function_atexit.htm. Accessed 18 Dec. 2022.

"Core Dump (Segmentation Fault) in C/C++." *GeeksforGeeks*, 28 Aug. 2017, www.geeksforgeeks.org/core-dump-segmentation-fault-c-cpp/.

Library, NYIT. "LibGuides: Raspberry Pi: SSH - Remote Login." *Libguides.nyit.edu*, libguides.nyit.edu/c.php?g=469894&p=3365470. Accessed 18 Dec. 2022.

"Raspberry Pi Documentation - Configuration." *Www.raspberrypi.com*, www.raspberrypi.com/documentation/computers/configuration.html.