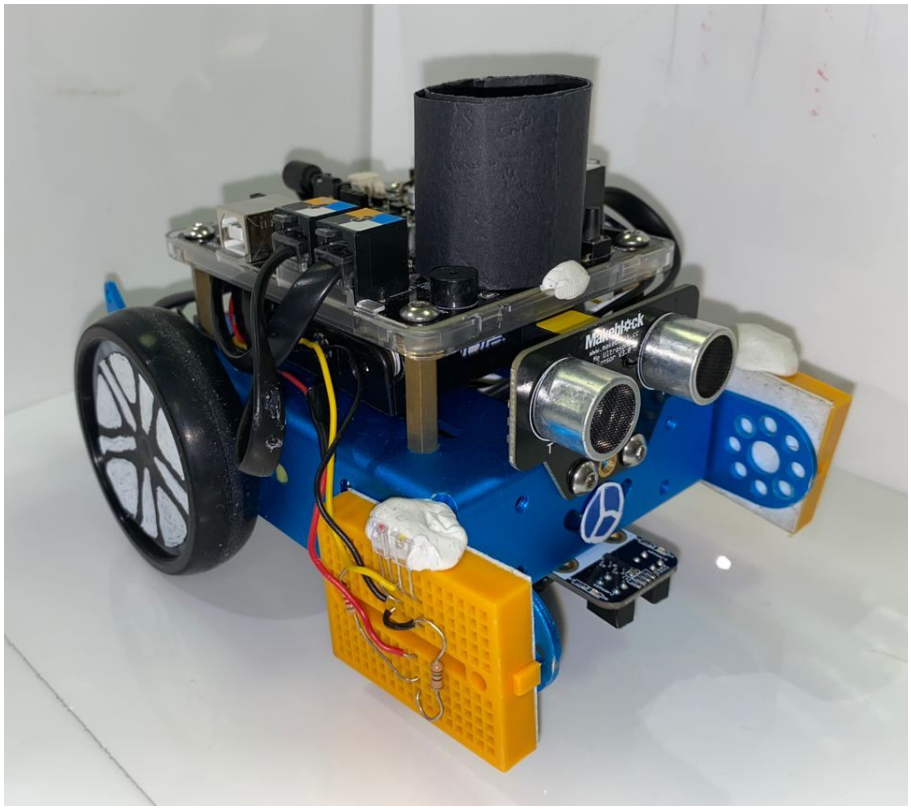
The A-maze-ing Race Project 2020 (20/21 Sem 1)



Group Number: 3A

Team Number: 4

Group Members:

Ng Andre, Mun Le Zong, Ngo Phuc Cuong,

Muhammad Ashraf Bin Mohamad Jaffar

Section 1 Overall Algorithm of mBot

Brief overview

The purpose of the project was to apply the principles we have learnt so far in CG1111 to build an mBot that can find its way through a maze in the shortest amount of time.

Outline of the Algorithm

As seen in Fig. 1 below, when the mBot is turned on, it starts moving forward until a wall or black line is detected. If a wall is detected, the mBot will correct its movement by turning slightly to the right or left away from the wall. When a black line is detected, the mBot stops. It will then activate its LED and read the colour above the mBot. Depending on the colour, the mBot will move in a given direction accordingly. If black is detected, signalling the end of the maze, the mBot will stop and play its victory tune.

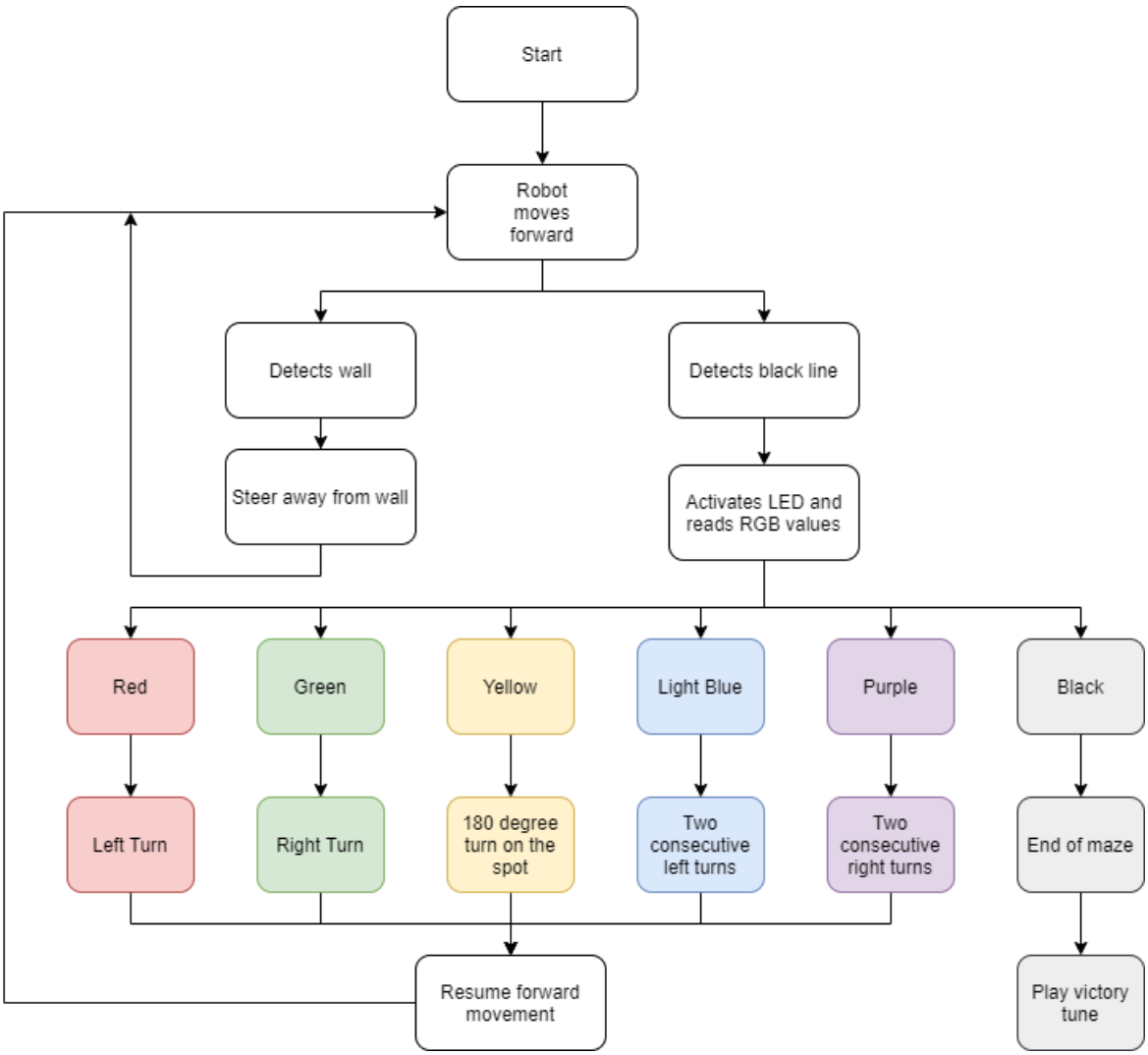


Fig. 1. Algorithm pseudocode

Section 2 Implementation of subsystems in mBot

Infrared Sensors

The set up for the left and right Infrared (IR) sensors are similar to the circuit we designed in Week 10 Studio 1 as seen in Fig. 2. The IR sensors consist of an IR Emitter and IR Detector in a potential divider circuit. By ensuring that both IR sensors have similar voltages, we can keep the mBot's movement straight. The left IR sensor is connected to A1, and the right IR sensor to A0 of Port 4 on the mCore. We then use the analogRead() function to read in the analogue inputs from both sensors. The values range from 0 to 1023 which represent voltage values from 0.0 to 5.0 V. As both sensors had different readings, we adjusted the threshold for each sensor individually. The mBot will then resume forward movement when the IR sensor reading is above the threshold, indicating that the path of the mBot is straight again.

When constructing the actual circuit, we opted to cut and use our own wires as seen in Fig. 3 below, instead of using the ribbon cables provided. This is to ensure that there will be minimal protrusions off the side of the mBot, so that the wires will be more secure and will not get disconnected if the mBot bumps into the walls on the side.

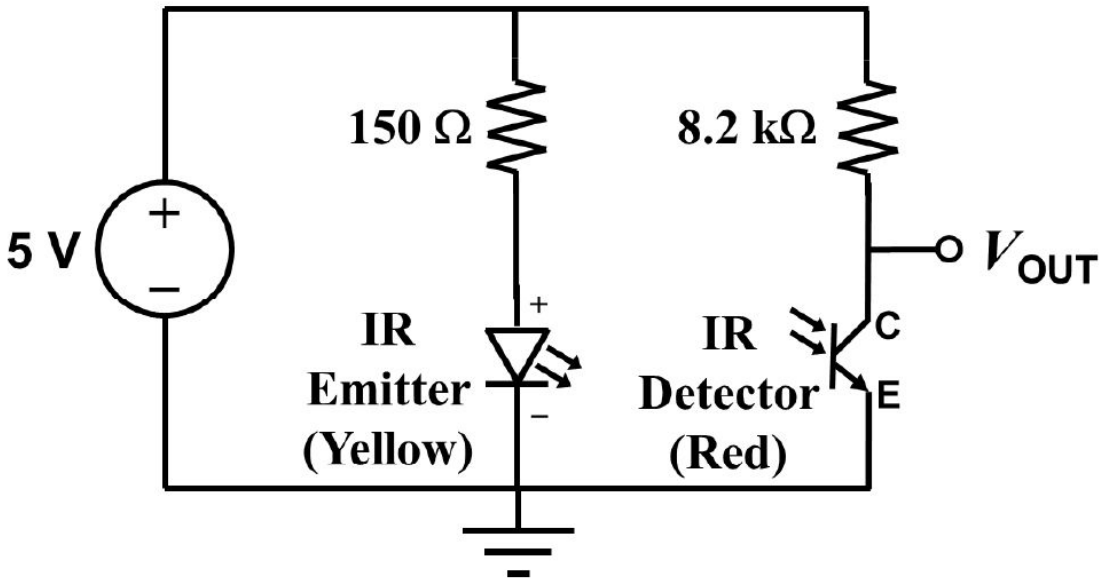


Fig. 2. IR Sensor Circuit [1]

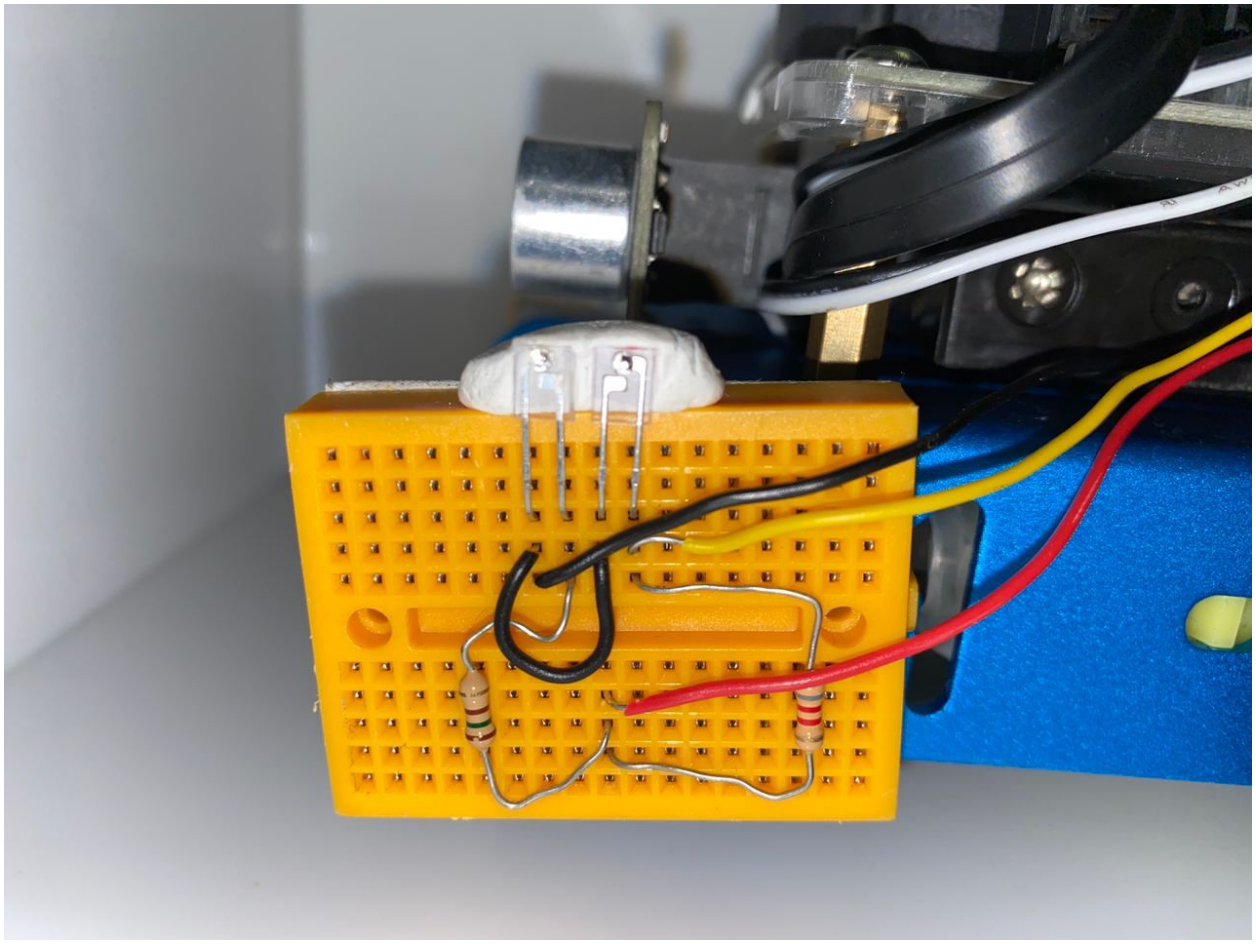


Fig. 3. IR Sensor Circuit as set up

Black Strip Detection

To enable the mBot to detect the black strip on the floor of the maze, we use the line Finder.readSensors() function within a Boolean function. The lineFinder function uses the 2 IR sensors connected below the mBot to detect the presence of the black strip. Whenever the black strip is detected by the sensors, the Boolean function will return true and calls for the colour sensor function.

Colour Sensor

The colour sensor function uses the RGB LED and the Light Dependent Resistor (LDR) attached to the mBot. The LDR's resistance varies inversely with the intensity of light. Since light is made of three primary colours red, green and blue, we can derive the colour above the mBot by reading the resistance values on the LDR for each of the three primary colours. This can be done because different coloured objects absorb and reflect different amounts of light. As such, by interpreting the data for each of the primary colours, we can deduce the colour above the mBot. For example, if the colour paper above the mBot is red, the LDR reading will be high when the red light is flashed and low when green and blue lights are flashed. This is because

the red colour paper will reflect most of the red light away while absorbing the green and blue lights. Once we retrieve the 3 individual light readings, we then compare it with our calibrated values for each of the 6 colour papers. When the reading falls within a range, the mBot will execute the specific movement required at that colour challenge.

Motor

The mBot uses a similar motor for the left and right wheels. Due to the construction of the mBot, the left motor has to run in the opposite direction of the right motor in order for the bot to move forward.

Turn Execution

Single 90-degree turn

In order to carry out a single 90-degree turn, the right and left motors will run in the opposite direction for a fixed amount of time. The direction is determined by the colour detected above the mBot. This will allow the mBot to make a 90-degree turn on the spot while clearing the black strip.

180-degree turn

The mBot executes a single 180-degree turn (within the same grid) in a similar manner to the single 90-degree turn, with both motors running in the opposite direction for double the amount of time it takes to execute the single 90-degree turn.

Two consecutive 90-degree turns

To execute 2 consecutive 90-degree turns within 2 grids, the mBot first performs a single 90-degree turn. It will then take in readings from the ultrasonic sensor at the front of the mBot using the ultraSensor.distanceCm() function in order to determine when to execute the consecutive turn. When the distance between the mBot and the maze wall in front falls below 12 cm, the mBot will execute the second turn.

Buzzer

When the mBot detects the black colour paper above it, the buzzer will be enabled. For the buzzer to play music, there will be two parts: the array for the notes of the melody and the duration of each note. To calculate each note duration, we take 1 second (1000ms) divided by each of the duration values in the array. The code will then iterate through the array of the notes, and activate the buzzer, using buzzer.tone() function to play each note for its duration. To distinguish the notes, there is a delay before playing the next note. After the end of the loop, the music is stopped by using the buzzer.noTone() function.

Section 3 Steps taken for calibration

Turning

We have used trial and error to fine-tune the exact timings for both the 90-degree and 180-degree turns. While there may still be variance in the turning angle due to changes in battery voltage, the margin of error is small and can still be corrected by the IR sensors when the mBot resumes forward movement.

Movement

There were variations in the left and right motor, possibly due to manufacturing defects. Through trial and error, we determined the individual motor speed for the left (210) and right (200) motors such that the mBot can move forward without having to correct its movement too much. While the aim was to complete the maze as quickly as possible, the maximum speed (255) was not used as the IR sensor could not react fast enough to correct the mBot's path, thus resulting in collisions.

Colour Sensor

We sampled a white sheet of paper to set the white balance for calibration. We went through each of the RGB colours one at a time and recorded the maximum reading for each colour. We then defined the readings as WHITE_R, WHITE_G and WHITE_B. Similarly, for the black balance, we sampled a black sheet of paper and went through each of the RGB colours one at a time. This time, we recorded the minimum reading for each colour and defined the readings as BLACK_R, BLACK_G and BLACK_B. The white and black balance values of each RGB colour are used to find the grey difference of each RGB.

Formula to calibrate 3 inputs values: VALUE_R, VALUE_G, VALUE_B

$$\text{CALIBRATED\_R} = 255 \times \left(\frac{VALUE\_R - BLACK\_R}{WHITE\_R - BLACK\_R} \right)$$

$$\text{CALIBRATED\_G} = 255 \times \left(\frac{VALUE\_G - BLACK\_G}{WHITE\_G - BLACK\_G} \right)$$

$$\text{CALIBRATED\_B} = 255 \times \left(\frac{VALUE\_B - BLACK\_B}{WHITE\_B - BLACK\_B} \right)$$

After calibration, we collected the range for each colour sample. We scanned and recorded the individual RGB values from the serial monitor. For higher accuracy, we took down the range of the RGB values for the different configurations of the walls, since the walls affected the amount of light received at the sensor. We then created conditional statements to differentiate the possible colours presented within the maze.

IR Sensor

We sampled the different voltage values measured by the IR sensor over a range of distances for calibration. As shown in Fig. 4 below, the voltage reading of the IR sensor will decrease when the sensor was brought closer to the wall. Once we determined an appropriate distance for the mBot to start correcting its movement, we set that distance as the threshold for the voltage value. Further fine-tuning was done through trial and error by observing the mBot's movement inside the maze while watching the voltage reading on the serial monitor of the mBot.

As the left and right IR sensors were not similar, we had to set different thresholds for each of them. The left IR sensor was given a threshold of 480. When a reading is below 480, the mBot senses that it is near to the left wall of the maze. When this happens, the right motor stops moving while the left motor continues moving thus swerving the mBot to the right, preventing it from colliding with the left wall of the maze. As for the right sensor, it was given a threshold of 600. Similarly, when a reading is below 600, the mBot will stop the left motor while the right motor continues moving, straightening its path and avoiding collision with the right wall.

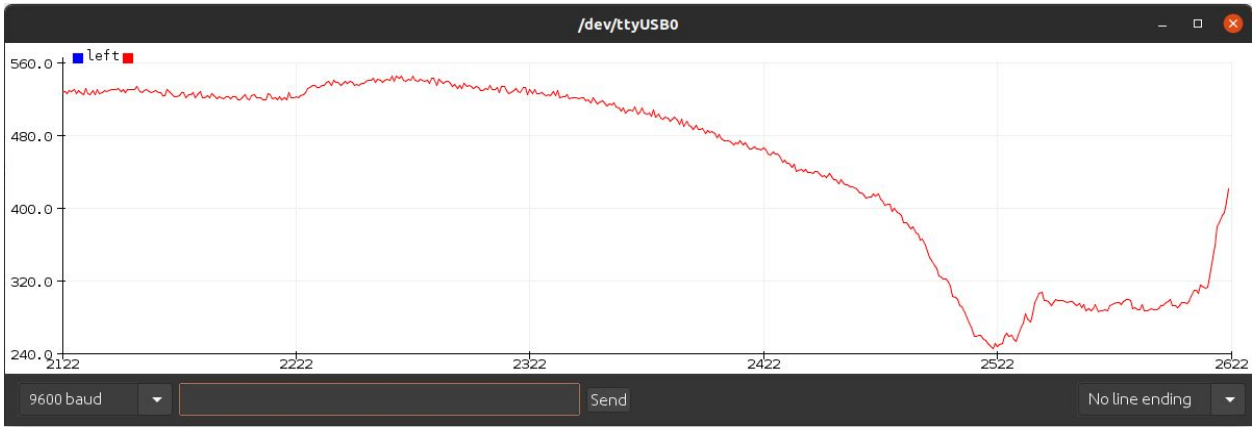


Fig. 4. Left IR Sensor Graph Plot

Ultrasonic Sensor

Through trial and error, we were able to determine the optimal distance for the mBot to move before making a second turn in the maze, such that it would not make contact with the walls when turning. This distance was determined to be 12cm, such that the mBot would be in the middle of the path when resuming movement after the turn.

Section 4 Work division

Each member of the group was tasked with a different subsystem to finish initially. However, this did not stop us from helping each other whenever problems surfaced. Eventually, we decided that it was better for all of us to work together on each subsystem since we managed to complete the task and debug much faster than working individually. At the same time, we believe that working together on each subsystem was better for our learning process since it enables us to gain experience on every component of the mBot as compared to implementing individual components. Additionally, we used an online coding platform (Repl.it) which allowed all 4 of us to code simultaneously. This was extremely helpful when we were calibrating the colour sensors since there were various conditional statements within the colour sensor function. Similarly for the report, we all contributed equally and worked at the same time so that we are all aligned on what is written.

Lastly, we would also like to give a special shout-out to Prof Henry, our TAs (Chin Tian and Ivander) and also Uncle Jalil (Lab Tech) for constantly providing us the help and guidance whenever we needed it.

Section 5 Challenges and actions taken

IR Sensor

One of the biggest challenges we faced during this project was getting the IR sensor to work properly. While setting up the circuit and reading in its voltage values was not an issue, getting the mBot to steer away from the wall took us a long time to accomplish. We faced many issues steering away when a wall was detected. The steering was eventually fixed by changing the code from slowing one motor to stopping the motor of the wheel on the opposite side from the wall, and by calibrating the threshold below which the turning function would activate. Initially, we faced an issue where the threshold would constantly fluctuate, giving us inconsistent performance. Eventually, we realised this was due to the position of the IR sensors changing every time it bumped into walls. Thus, we used Blu Tack adhesive putty to secure the IR sensors in place, which resulted in more consistent run-to-run readings.

Colour Sensor

Another challenge we faced was getting colour detection calibration to work under different configurations of the surrounding walls. The number of surrounding walls affect the amount of ambient light. If the mBot is in a dead-end corner with 3 walls, the amount of ambient light is very low. If the mBot is in a right turn with only 1 wall, the amount of ambient light is very high. We originally calibrated our colour detection subsystem with a reasonable number of surrounding walls. For instance, if it’s a right turn, we assume that there are 2 surrounding walls

- 1 to the left and 1 in front. If it's a 180-degree turn, we assume that there are 3 surrounding walls. We set up the wall configuration for each colour and calibrated our mBot. However, during the test run, we noticed that the number of surrounding walls could vary a lot depending on the different map configurations that would be decided by the professor. As such, we needed to adjust the detection range for each colour so that our mBot could detect a colour even when it was too bright and too dark. We came up with a quick and effective fix. We decreased the lower bound of the range to detect colours in dark condition and increased the upper bound of the range to detect colours in bright condition. This worked well for us because the same colour in bright condition will have higher RGB value than in normal condition and the same colour in dark condition will have lower RGB value than in the normal condition.

Battery Voltage

The third challenge we faced was with the changing voltage of the battery. As the batteries on the mBot discharged, the voltage supplied by the batteries would decrease over time. These changes in voltage supplied would affect many subsystems, such as the motor and the IR sensor. As the voltage dropped, the motor speed would decrease, thus the time delay for turn execution would vary from run to run. Also, as the IR sensor is effectively a potential divider circuit, when battery voltage decreases, the voltage across the IR sensor would not remain constant. This resulted in a run to run inconsistencies and caused the calibrations for the other subsystems to vary. In order to mitigate such an issue, we took voltage measurements at the terminals of the breakout board when the batteries were at full charge and based our calibrations at that voltage. Once the voltage had dropped beyond an acceptable level, the batteries would have to be charged, allowing us to calibrate the other subsystems properly.

Section 6 References

[1] H. Tan, "Photoelectric Sensors, CG1111 Week 10 Studio 1," unpublished.