CG1111 MBOT PROJECT

GROUP 1A–5

A-maze-ing Race Project

Prepared by:

Chong Xuan Liang, Chew Yi Jie, Christopher Langton, Christopher Nge

SCOPE

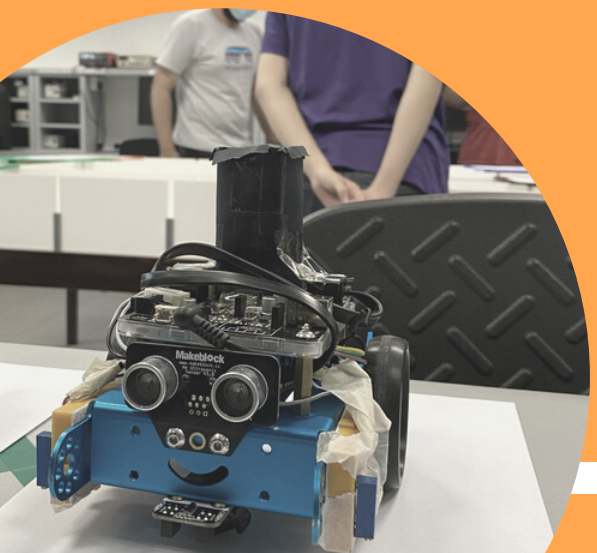
01 Overall Algorithm

02 Implementation of Subsystems

03 Calibration of Subsystem

04 Workload Division

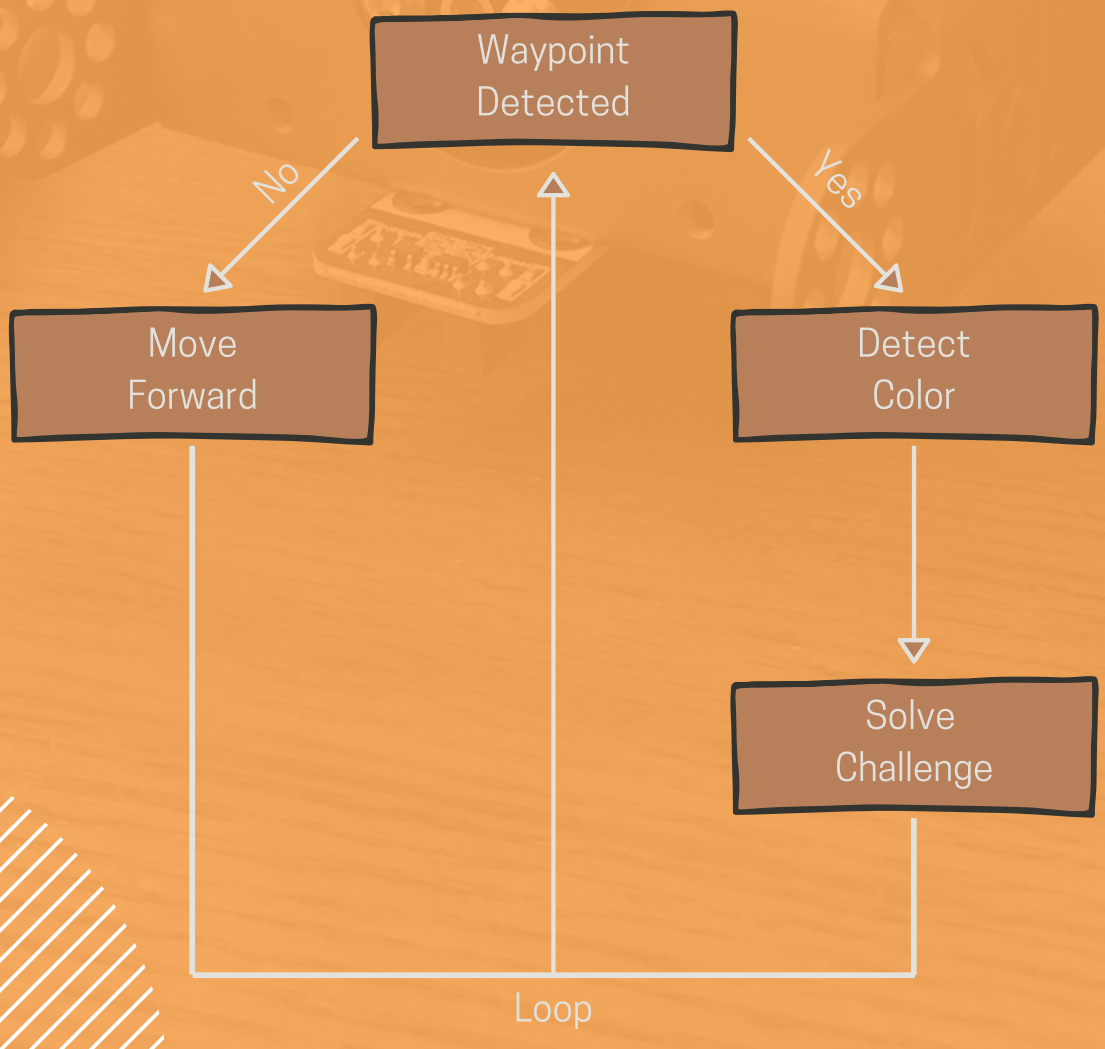
05 Challenges & Actions Taken



Gary the ScrapBot

01 Overall Algorithm

Overview of main loop



02 Implementation of Subsystems

This section outlines the various subsystems of the mBot which include sensors, motion, color detection, and waypoint challenge solving.

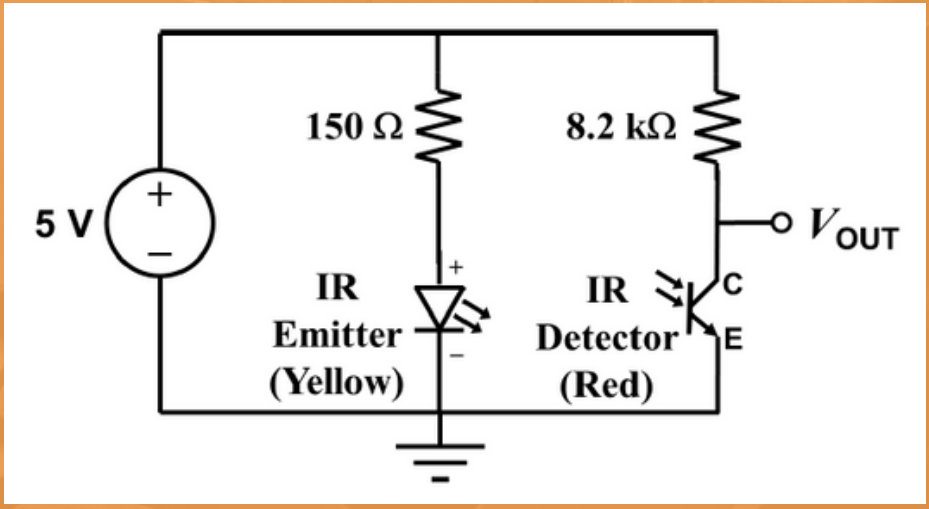
Sensors

The mBot is affixed with 3 types of sensors, namely Infrared proximity sensors, ultrasonic sensor, and a line detector.

Infrared sensors

The IR sensors are attached to the right and left of the mBot to detect the distance from the side walls. The readings from each sensor are used to adjust the angle of the mBot's movement to prevent collision against side walls.

The circuit layout is as follows:



The sensors are connected to the mBot via an RJ25 adaptor.

Function	Description
void onIR()	Turns on IR sensors
int readLeft() int readRight()	Returns the IR sensor reading on corresponding side

Infrared sensors

An ultrasonic sensor is attached to the front of the mBot to detect the distance from the front wall.

Function	Description
double getFrontDistance()	Returns distance from a front wall

Line detector

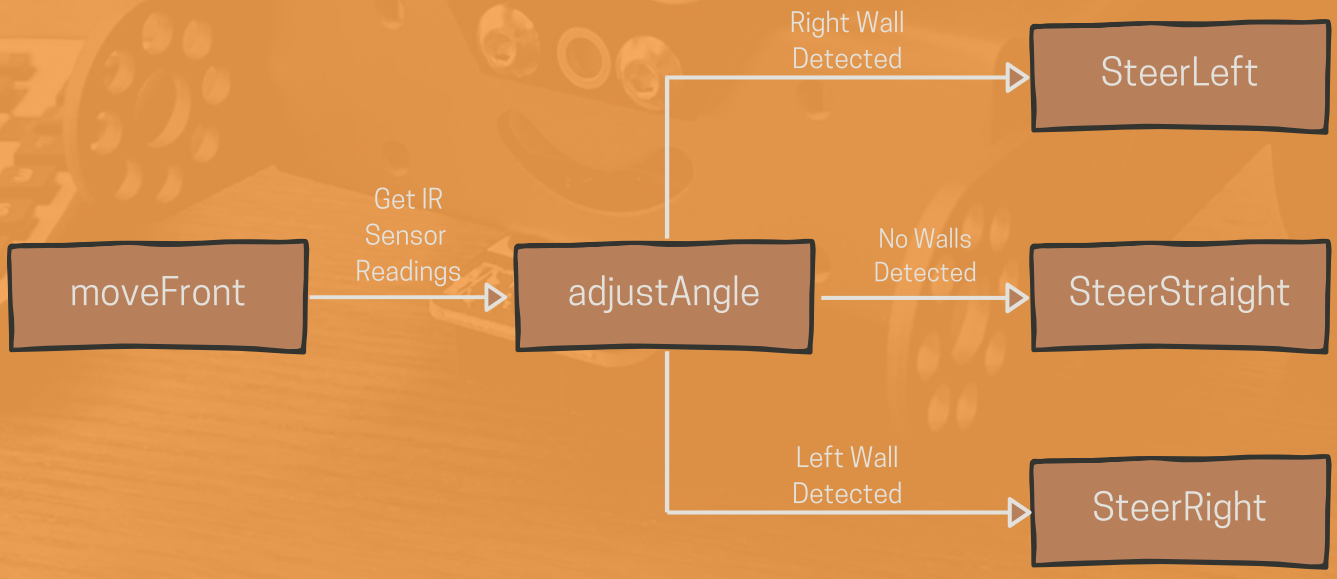
Due to the construction of the line detector (2 sensors) it can be determined if the mBot approaches a waypoint at an angle. This allows the mBot to correct its orientation such that it will be perpendicular to the line.

Function	Description
bool atWaypoint()	Checks if a line is detected. If only one side of the line detector detects the line, the mBot then runs adjustWaypointLeft/Right function described in “Motion” subsection to correct itself

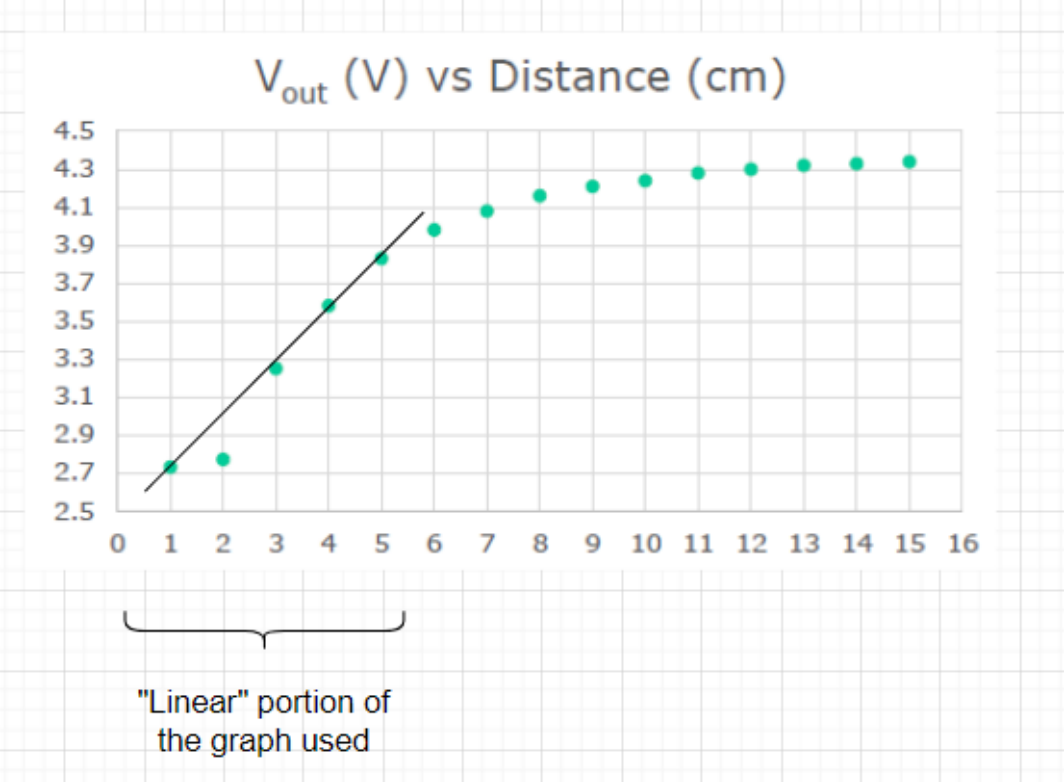
Motion

The Movement of the mBot is determined via values read from the IR sensors. Due to the short range of the IR sensors, the mBot is steered away from a detected wall in order to keep a fixed distance from both walls. The steering angle is determined by the distance of the mBot from each wall calculated through the linear portion (roughly 0 to 5cm) taken from each IR sensor's distance-voltage curve. This ratio then determines the speed of each motor, so that the mBot will steer according to how close it is to a wall to prevent over/under-steering.

Overview of Motion



02 Implementation of Subsystems



Functions

Side wall avoidance motion

Functions	Description
void moveFront()	Begins forward motion of the mBot
void adjustAngle(int right, int left) Parameters left (int): analog read value of left IR sensor right(int): analog read value of right IR sensor	Based on IR sensor readings, mBot adjusts it direction accordingly
bool hasLeftWall(int left) bool hasRightWall(int right) Parameters left (int): analog read value of left IR sensor right(int): analog read value of right IR sensor	Checks if mbot is close to corresponding wall with reference calibrated values
void steerLeft(int right) void steerRight(int left) Parameters left (int): analog read value of left IR sensor right(int): analog read value of right IR sensor	Steers the mbot by reducing speed on one wheel distance to a wall is estimated through the distance-voltage graph of each IR sensor and the current to max-distance ratio determines how hard to steer
void steerStraight()	No steering, full speed ahead
void halt()	Stops mbot

Waypoint correction motion

Functions	Description
void adjustWaypointLeft() void adjustWaypointRight()	Turns the mBot until the line is detected by both sides of the line detector

Waypoint challenge motion

Functions	Description
void turnLeft() void turnRight()	Turns the mBot 90 degrees
void uTurn()	Turns the mBot 180 degrees. The direction of the rotation is determined by the presence of any side walls. The mBot will rotate away from the nearest wall, if present
void doubleLeft() void doubleRight()	Turns, then moves forward until a front wall is at a defined distance from the ultrasonic sensor, where it then turns again

Color Detection

The built in LEDs are controlled through pin 13 of the mBot.

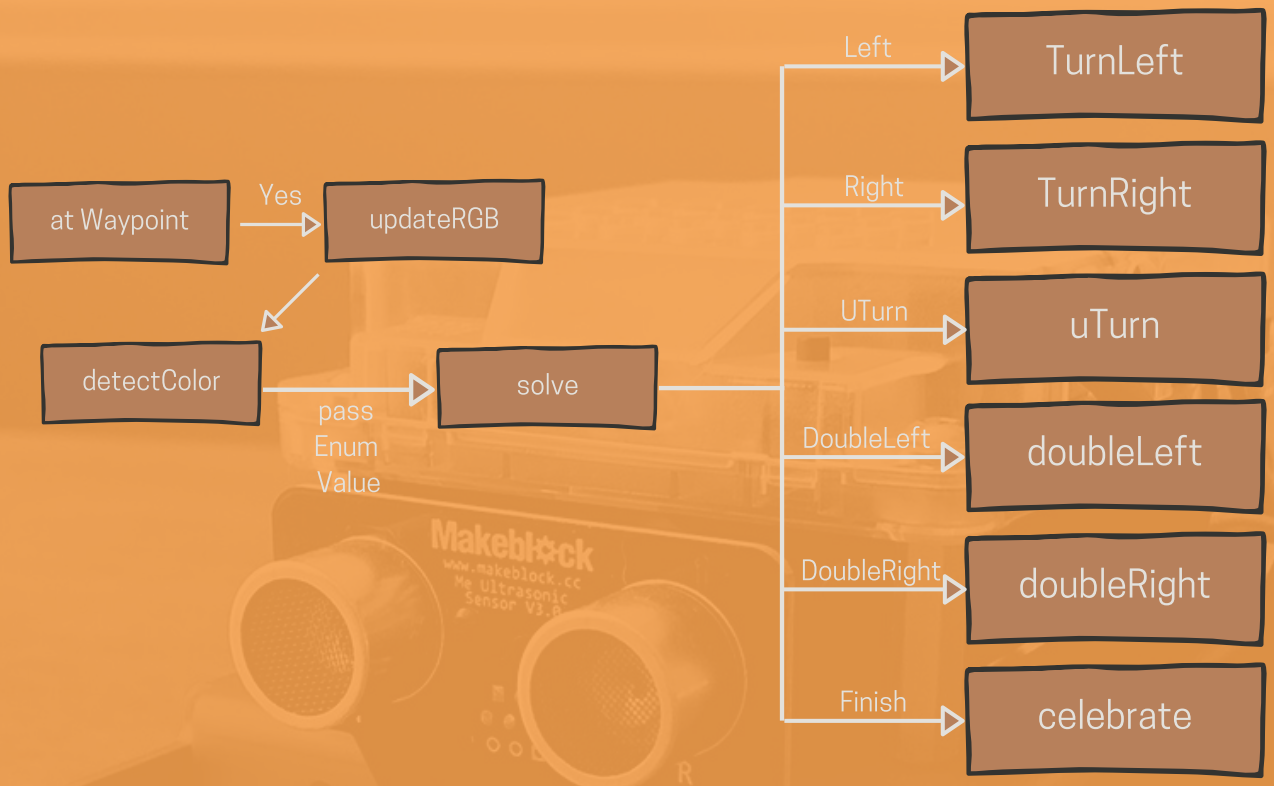
A chimney is placed over the LEDs and color detector in order to reduce interference from ambient lighting. Through a series of led flashes, the average RGB values are recorded to determine the detected color.

Each color is assigned to an enumeration value of type sign for readability.

Colour	Enum Value
Red	LEFT
Green	RIGHT
Yellow	UTURN
Purple	DOUBLE_LEFT
Light Blue	DOUBLE_RIGHT
Black	FINISH

Functions	Description
void updateRGB()	updates the average RGB values of the detected color
Sign detectColor()	Using the detected RGB values, return the enum value for the detected color

Waypoint Challenge Solving



From the enum received from color detection, a simple switch case is implemented to run the appropriate solution.

For the case where the sign is FINISH, the 'celebrate' function is run instead of the usual motion functions.

Functions	Description
void solve(Sign sign) Parameters sign (Enum Sign)	Switch case to decide which solution to run
void celebrate()	Stops the mBot and plays a celebratory tune through the buzzer. An infinite loop is then entered to prevent any further actions

03 Steps Taken for Calibration

The major calibrations performed for this project were for movement as well as for the ultrasound, IR and color sensors.

3.1 Movement:

The turns of the mBot were calibrated to ensure that the 90 degrees and 180 degrees for the right angle turns and U-turns respectively. Turning is done on the spot to avoid collision with any walls.

3.2 Ultrasound Sensor:

The purpose of the ultrasound sensor is to obtain the distance of the mBot from a wall in front of it. This is only used in the event of either a double right or double left turn, to let the mBot know when to execute the second turn. The mBot would turn at a fixed distance of 10cm from the front wall. This distance was calibrated so that the mBot would turn into the centre of the path.

3.3 IR Sensor:

The purpose of the IR sensors is to get the mBot to steer away from the walls, based on the sensor readings. The IR sensors of the mBot were calibrated by plotting values of the IR sensor readings against distance from the wall in Excel. By using the graph equation, to scale the magnitude of the turning of the mBot based on how close it is to a wall, instead of using a fixed turning magnitude. This allowed the mBot to avoid zigzagging when avoiding walls.

3.4 Color Sensor:

The purpose of the color sensor is to identify the color above the mBot during the color challenge. After recalibrating the color sensor multiple times, it was observed that although the values changed based on battery level, the values of each color relative to black remained near constant. Hence, the color sensor is calibrated with the battery at full charge for all tested colors. The calibrated value of black were subtracted from the read values and compared with the resulting values to our calibrated color values to identify the color.

04 Workload Division

Roles and Responsibilities

Chong Xuan Liang

- Creator and contributor to framework of code for each of the functioning subsystems of the mBot.
- Main contributor to motion and proximity codes run by the mBot.
- In charge of adjustments in linear motion and execution of turns of the mBot throughout the testing stage.
- Assisted in the adjustments in IR sensor calibration of the mBot during the testing stage.

Chew Yi Jie

- Main contributor to the color code run by the mBot.
- In charge of adjustments to Color Calibration of the mBot throughout the testing stage.
- Assisted in the adjustments in IR sensor calibration of the mBot during the testing stage.

Christopher Nge

- Main contributor to the waypoint and puzzle code run by the mBot.
- In charge of the making and placement of functional 'chimney' housing the LEDs to assist in greater color detection accuracy of the mBot.
- Assisted in the colour calibration of the mBot throughout the testing stage.

Christopher Langton

- Main contributor to the tune code run by the mBot.
- In charge of the making and placement of the IR Sensors for side wall detection on either side of the mBot.
- Assisted in the adjustments in IR sensor calibration of the mBot throughout the testing stage.

05 Challenges and Actions Taken

Challenge:

Accuracy of colour detection deteriorates over time as the draining of the rechargeable battery affects the voltage supply to the light sensor, giving decreased readings from the sensor. This causes the readings to deviate from the initial calibration, such that they eventually fall outside the range of values that corresponds to the correct colour. This results in the incorrect detection of color, subsequently causing the mBot to make a wrong turn.

Action Taken:

First, our team hard-coded the values from the light sensor for color calibration at full charge of the batteries, such that each color's range of values is set for operation at full battery capacity. Subsequently, for every test run of the mBot and most importantly for the final project evaluation, our team ensured the mBot operated with the batteries at an initial charge of 100% to minimise the deviation from calibrated values and hence mitigate the risk of incorrect color detection. Our Team took the additional measure of acquiring an extra set of rechargeable batteries and charging them to full capacity, in the event of a failure in the first set of batteries.

Challenge:

The initial positioning of the IR sensors on either side of the mBot caused their orientation to change dramatically upon collision of the mBot with a side wall. This change in orientation of the IR sensors would disrupt the readings taken by them, and hence subsequently affect the capacity of the mBot to correct its linear motion. As collisions occurred quite frequently during testing, this issue resulted in our team needing to repeatedly recalibrate the IR sensors after each collision. This would also pose an issue if the mBot encountered a collision during the final evaluation, as its linear motion would be compromised.

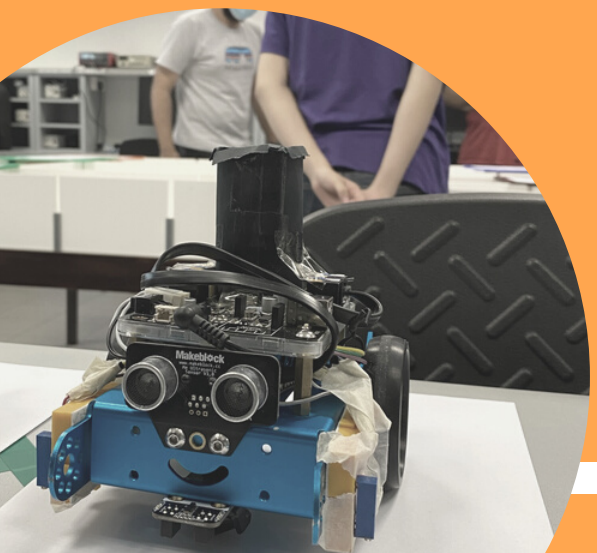
Action Taken:

Our team decided to shift the placement of the breadboard towards the centre of the mBot, as well as tape in place a dummy variable resistor at the front-most row of the breadboard to act as a shield for the IR sensors in the event of a collision, as well as blue-tack the IR sensors firmly in their position on the breadboard. After this measure was taken, the IR sensors faced no recalibration issues whatsoever upon multiple collisions during testing as well as during the final evaluation, allowing the mBot to carry out smooth linear motion.

FINAL WORDS

The A-maze-ing Project is truly an apt project that has adequately summarised the learnings of CG1111. From the construction of the code to the placement of the sensors, it epitomises the beautiful relationship between hardware and software. Though there were many seemingly insurmountable problems faced along the way, through careful planning and problem solving, our team was able to overcome them and emerge with pride in the final product of our efforts

As the module draws to a close, as a group we share the similar sentiment that the very essence of this project, communication between software and hardware, serves as a microcosm of our communication with one another as teammates.



Gary the ScrapBot Signing off