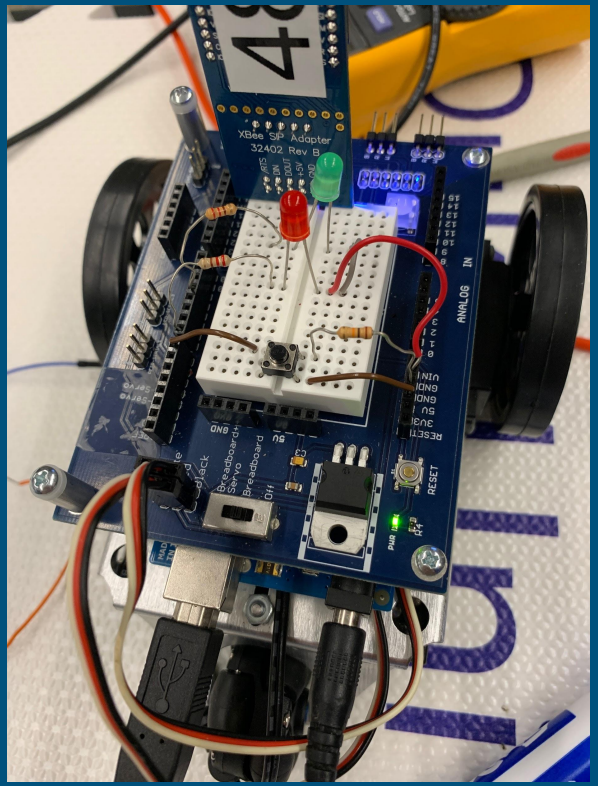
# IDC Weekly Status Update 2 (10/31/19)

Tracking Tags Team: Josh Boss and Jason Dong

# Progress Summary - Successes

- Implemented Line Following
  - Bot can move in various shapes - square, curved line, hashed line, circle
- Implemented RFID Sensing Module
  - Detects RFID Chip and lights up an LED in response



# Progress Summary - Challenges

- Complications when trying to combine Line Following with Communication and Sensing
- Finding a way to hook up RFID Sensor onto bot
- Accidently shorted RFID Sensor

```
#include<Servo.h>


Servo servoLeft;
Servo servoRight;

int QTIPinL = 47;  //initialize qti pinL variable for input pin
int QTIPinM = 51;   // ^
int QTIPinR = 52;   // . ^
int QTIVal_L = 0;  //Sets the reading from QTI to 0
int QTIVal_M = 0;   //Sets the reading from QTI to 0
int QTIVal_R = 0;     //Sets the reading from QTI to 0
int Threshold = 200;  //Set's threshold between black and white to 200
void setup() {
  Serial.begin(9600);  //Initializes serial monitor
  //initialize servo
  servoLeft.attach(12);    //initilizes left wheels
  servoRight.attach(11);   //initializes right wheels
  servoLeft.writeMicroseconds(1500);    //make sure it is stationary
  servoRight.writeMicroseconds(1500);
}
```

```
void loop() {
  // different if statements based on what is seen
  // compare to threshold
  // change servo speed accordingly

  QTIVal_M = rcTime(QTIPinM);   //obtain rcTime for middle sensor and store in variable
  QTIVal_L = rcTime(QTIPinL);   //obtain rcTime for left sensor and store in variable
  QTIVal_R = rcTime(QTIPinR);   //obtain rcTime for right sensor and store in variable


  //PRINTS IF ITS BLACK OR WHITE
  if(QTIVal_L <= Threshold) Serial.println("white");
  if(QTIVal_L >= Threshold) Serial.println("black");
  if(QTIVal_M <= Threshold) Serial.println("white");
  if(QTIVal_M >= Threshold) Serial.println("black");
  if(QTIVal_R <= Threshold) Serial.println("white");
  if(QTIVal_R >= Threshold) Serial.println("black");
  Serial.println(" ");


  if(qtiLogic( QTIVal_L,  QTIVal_M,  QTIVal_R) == 0){ //if qti logic function says 0, execute corresponding action
    goForward();                                      //calls goForward function below
    }

  else if(qtiLogic( QTIVal_L,  QTIVal_M,  QTIVal_R) == 1){ .  //if qti logic function says 1, stop, then move foward
    servoStop();      //calls stop function
    delay(2000);      //wait for 2 seconds
    goForwardHash();   //call go foward hash function
    delay(100);     //weight .1 seconds
    }

  else if(qtiLogic( QTIVal_L,  QTIVal_M,  QTIVal_R) ==2){ .  //if qti dictates, go left
    turnLeft();
    }

  else if(qtiLogic( QTIVal_L,  QTIVal_M,  QTIVal_R) == 3){ .  //if qti dictates, go right
    turnRight();
    }
```

# Code - Line Following

- Top Left picture initializes all of our variables, including the pins connected to QTI sensors to the bot.
- It also sets the black/white threshold to 200 (which we found via experimentation)
- Show's our setup, which initializes serial monitor so that we can have print statements to check what sensors are reading, and initializes the right and left wheels via the servo methods. Makes sure they are initially at rest

- Bottom Left picture begins loop, which calculates the rcTime of what QTI pins are reading for each of the three sensors, and then prints it out in the serial monitor
- Begins logic for QTI sensors,  calls qtiLogic function, and depending on return value tells car to move forward, left, right, or stop.

```
else if(qtiLogic( QTIVal_L,  QTIVal_M,  QTIVal_R) == 3){ .  //if qti dictates, go right
  turnRight();
  }


}
```

```
int qtiLogic(int QTIVal_L, int QTIVal_M, int QTIVal_R){ .
    // 0 - GO STRAIGHT
    // 1 at long hashmark (stop points) - STOP
    // 2  TURN LEFT
    // 3  TURn RIGHT
    // 4 - ID KNOW SOMETHING WENT WRONG

  if( QTIVal_L >= Threshold && QTIVal_M >= Threshold && QTIVal_R >= Threshold){ . //if all three sensors black, stop
    return 1;
    }

  if( QTIVal_L >= Threshold && QTIVal_M >= Threshold && QTIVal_R <= Threshold){ . // if left and middle black, turn left
    return 2;
    }

  if( QTIVal_L >= Threshold && QTIVal_M <= Threshold && QTIVal_R <= Threshold){ .  //if just left black, turn left
      return 2;
    }


  if( QTIVal_L <= Threshold && QTIVal_M >= Threshold && QTIVal_R >= Threshold){ . //if right and middle black, turn right
      return 3;
    }

  if( QTIVal_L <= Threshold && QTIVal_M <= Threshold && QTIVal_R >= Threshold){ . //if just right black, turn right
      return 3;
    }

  if( QTIVal_L <= Threshold && QTIVal_M >= Threshold && QTIVal_R <= Threshold){ . //if middle is only black, move foward
      return 0;
    }

  if( QTIVal_L <= Threshold && QTIVal_M <= Threshold && QTIVal_R <= Threshold){ .  //No logic needed for this case
      return 4;
    }

}
```

# Code - Line Following

- Top Left picture finishes QTI logic calls

- Bottom Left picture shows the QTI logic function.
- Depending on what the sensors are reading in relation to threshold value, it determines whether the car should turn left right, forward, or stop. Each if-statement returns a number corresponding to one of the above actions. For example if the right sensor is reading black, but the other two are reading white, the cart should turn right to get back onto the line.

```
void goForward(){ .        //Function tells servo to move foward
  servoLeft.writeMicroseconds(1600);
  servoRight.writeMicroseconds(1400);
}

void goForwardHash(){ .  //function tells to move foward after being stopped
  servoLeft.writeMicroseconds(1700);
  servoRight.writeMicroseconds(1300);

}

void servoStop(){   //function tells servo to stop
  servoLeft.writeMicroseconds(1500);
  servoRight.writeMicroseconds(1500);
}

void mainTurn(){ .   //function tells servo to turn - not used
  servoLeft.writeMicroseconds(1750);
  servoRight.writeMicroseconds(1350);
}

void turnLeft(){                      // Left turn function
  servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
  servoRight.writeMicroseconds(1300);       // Right wheel clockwise
}

void turnRight(){                      // Right turn function
  servoLeft.writeMicroseconds(1700);        // Left wheel counterclockwise
  servoRight.writeMicroseconds(1700);       // Right wheel counterclockwise
                                        // Maneuver for time ms
}
```

```
long rcTime(int pin) { .  //RCtime - converts reading from qti into value
 pinMode(pin, OUTPUT);
 digitalWrite(pin, HIGH);
 delayMicroseconds(230);
 pinMode(pin, INPUT);
 digitalWrite(pin, LOW);
 long time = micros();
 while (digitalRead(pin));
 time = micros() - time;
 return time;
}
```

# Code - Line Following

- The top left picture is the collection of functions that direct the cart to move the way it should. For example if the qti readings dictate that the cart should move left, code from above calls one of these functions, and then this function directs the car to do just that, via servoLeft and servoRight .writeMicroseconds() function.

- Bottom left picture is the rcTime function, which converts sensor reading to an rc Time. The loop is constantly updating these values to ensure the car stays over the line.

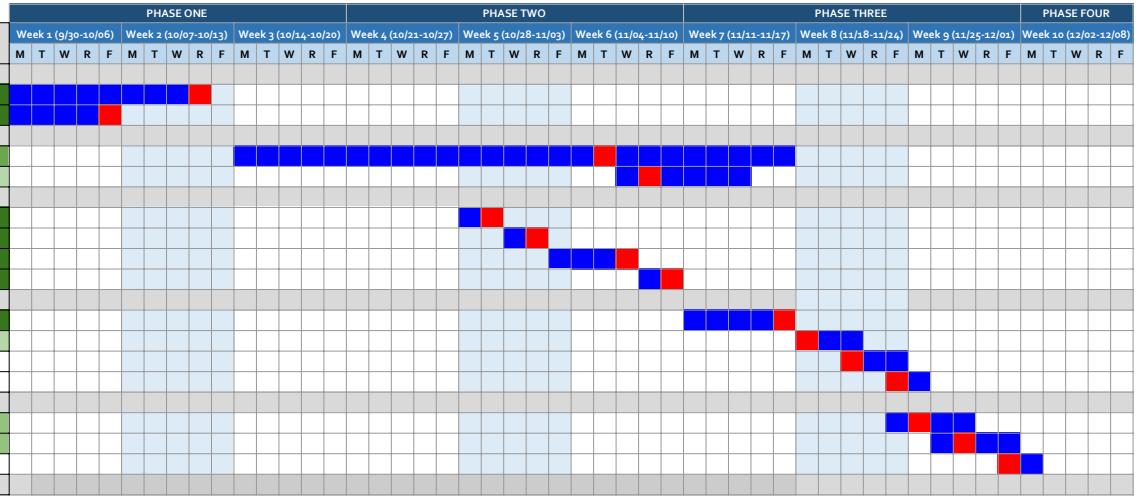
```
char val = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);   //initialze serial
  pinMode(49, INPUT);    //set pin49 as input
  pinMode(3, OUTPUT);   //set pin 3 as output
  digitalWrite(3, LOW);   //start will LED off


}

void loop() {
  // put your main code here, to run repeatedly:
  if(rcTime(49) != NULL){ .  //if rctime function call is yeilding value
    Serial.println(rcTime(49));  //print out that value
    digitalWrite(3, HIGH);   //turn LED on
    delay(3000);             //wait 3 seconds
    digitalWrite(3, LOW);    //turn it off
    //delay(2000);            /wait 2 more seconds
  }

}

long rcTime(int pin) { .    //RCtime function returns value if
 pinMode(pin, OUTPUT);       //RFID is read
 digitalWrite(pin, HIGH);
 delayMicroseconds(230);
 pinMode(pin, INPUT);
 digitalWrite(pin, LOW); long
 time = micros(); while
 (digitalRead(pin));
 time = micros() - time;
 return time;
}
```

# Code - RFID Sensing

- The Code to the left is a sketch that is used to detect whether or not an RFID tag is present.
- In the setup, we initialize serial monitor, pin 49 (RFID reader) as an input, and pin 3 (wired to LED) as an output
- In the loop, we check to see if the rcTime function for the RFID reader is indicating that a tag is present, and if it is, we turn the light on for three seconds, then turn it off.
- The rcTIme function gives us a value for the RC Time if there is a tag present. If there is no tag present, then it doesn't give us a value.

NOTE - communication code is same as last week

| Task Number | Task Title | Team Members Involved | due Date | Percent of Task Completed - By Josh | Percent of Task Completed - By Jason | Percent of Task Completed |
|---|---|---|---|---|---|---|
| 1 | Project Conception and Initiation | | | | | |
| | Understand our Task and System/Plan | | 9/30/19 | 50% | 50% | 100% |
| | Conceptual Design Report 1 | | 10/2/19 | 45% | 55% | 100% |
| 2 | Communication | | 10/14/19 | | | |
| | Utilize XBEE Sensor | | | 32.5 | 32.5 | 75% |
| | Displaying Value | | | 12.5 | 12.5 | 25% |
| 3 | Line Following | | 10/28/19 | | | |
| | Bot Movement | | | 50% | 50% | 100% |
| | Implement Sensors | | | 50% | 50% | 100% |
| | Line Following | | | 50% | 50% | 100% |
| | Stopping | | | 50% | 50% | 100% |
| 4 | RFID Sensor Installation and Stroring Data | | 11/11/19 | | | |
| | Implement Sensors | | | 50% | 50% | 100% |
| | Sensor positioning | | | 13% | 13% | 25% |
| | Data Storage | | | | | 0% |
| | Processing Data | | | | | 0% |
| 5 | Team Integration | | 11/18/19 | | | |
| | Outward Communication | | | 25 | 25 | 50% |
| | Input reading | | | 25 | 25 | 50% |
| | Perform Team-Coordinated Response | | | | | 0% |
| 6 | Oral Design Explanation and Defense | | 12/12/19 | | | 0% |



Cost of BOT:

- The RFID Module is our only additional sensor as of right now, and costs $29.95. We have not submitted requests for any other parts yet, and only anticipate using materials from lab, such as a 7-segment display, and LEDs.

Adding up additional parts from our BOE-Bot up to completing communication:

- 2x BOE-Bot plastic wheel with tire ($4x2) = $7.98
- 1x BOE-Bot tail ball wheel = $3.95
- 1x BOE-Bot Aluminum Chassis = $24.99
- 1x BOE-Bot Li Ion Power Pack with cable and barrel plug = $49.99
- 1x Li Ion Cell = $8.99
- 1x 3/8" x 2" (5.1 x 3.5 cm) solderless breadboard = $3.49
- 2x Standard Servo Motor ($12.99 x 2) = $25.99
- 1x Arduino ATMEGA 2560 $51.91
- 1x Board of Education Shield for Arduino $39.99
- 1x USB A to B Cable = $4.99
- 1x 7.5v 1A power supply = $14.99
- 1x XBee Module = $22.99
- 1x RFID Module = $29.95
- 4x 3/8" 4-40 pan head screw (each) (4x0.02) = $0.08
- 4x nylon washer (screw size #4) (4x0.07) = $0.28
- 2x LED (1 Red, 1 Green) (2x0.32) = $0.64
- 1x Push button tact switch = $0.50
- 2x 220 Ohm 1/4 W resistor (2x0.10) = $0.20
- 10 kOhm 1/4 W resistor = $.0.10
- Wire, 22 AWG, solid, 100 ft,Blk ($0.08/ ft.) / 6 inches used = $0.04
- Arduino Wiring Kit - $9.95

Total Estimated Cost To Date: $281.81

Updated 10/27/19 - No additional parts expected currently, bot cost remains the same.