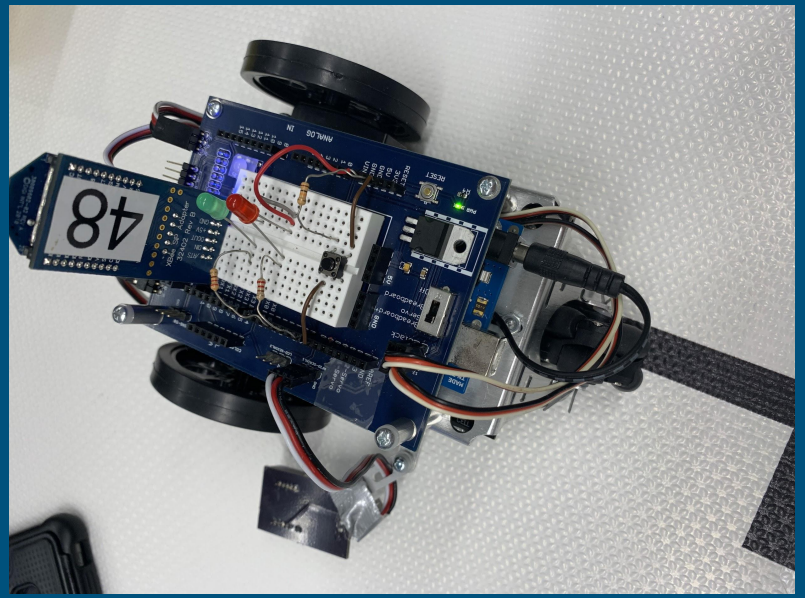
IDC Weekly Status Update 3 (11/14/19)

Jason & Josh
RFID Sensing Group

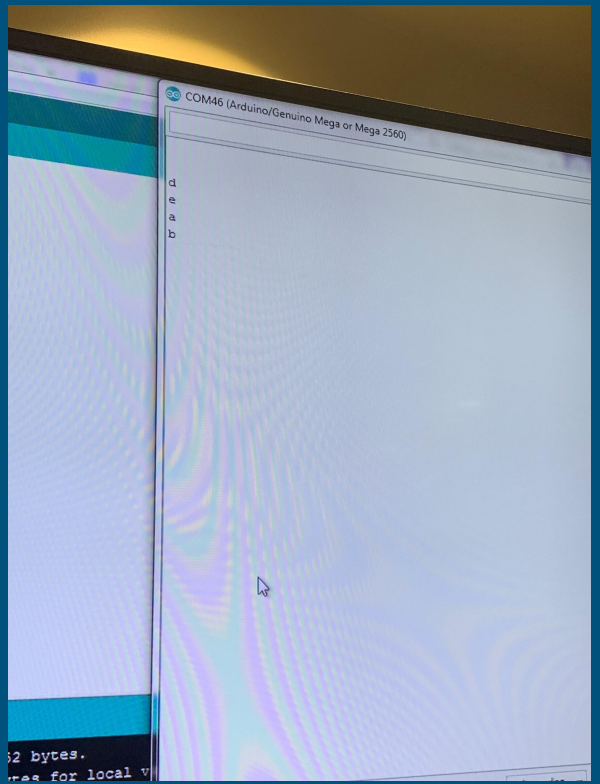
Progress Summary - Successes

- Successfully implemented hash counting and stopping after 5th hash
- Capable of sensing after stopping at each line and storing the result in counter
- After stopping on the 5th hash, report the results with class XBee



Progress Summary - Challenges

- Some difficulties at first integrating everything together
- Spent time trying to communicate numerical values via ASCII table



Code

```
#define BUFSIZE    15  // Size of receive buffer (in bytes) (10-byte un
#define RFID_START  0x02  // RFID Reader Start and Stop bytes
#define RFID_STOP   0x03
int RFIDcounter = 0;
bool isThere = false; //
bool shouldPrint = true;


#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
int HashCounter = 0;
```

```
void setup() {
  //From comminication
    Serial2.begin(9600);
    Serial1.begin(9600);                          // Inintialize the RFID serial port




  Serial.begin(9600);  //Initializes serial monitor
   Serial.flush();                          // wait for all bytes to be transmitted to the 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);
}
```

- Initializes variables
- Sets up servo and serials

```
void loop() {
  //from RFID
  char rfidData[BUFSIZE];                       // Buffer for incoming data
  char offset = 0;                                                  // Offset into buffer
  rfidData[0] = 0;


  // 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


  if(HashCounter <5){

  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
    HashCounter++;
//    Serial.print(HashCounter);
    delay(2000);      //wait for 2 seconds

    while(Serial1.available() > 0)                      //Loop that waits for a tag to be read
  {


    if (Serial1.available() > 0)             // If there are any bytes available to read, then the RFID Reader has probably seen a valid tag
    {
      isThere = true;
      Serial.print(isThere);
      rfidData[offset] = Serial1.read();        // Get the byte and store it in our buffer
      if (rfidData[offset] == RFID_START)    // If we receive the start byte from the RFID Reader, then get ready to receive the tag's unique ID
      {
        offset = -1;     // Clear offset (will be incremented back to 0 at the end of the loop)
      }
      else if (rfidData[offset] == RFID_STOP)  // If we receive the stop byte from the RFID Reader, then the tag's entire unique ID has been sent
      {
        rfidData[offset] = 0;                     // Null terminate the string of bytes we just received
        //COMUNICATION CODE


        break;                                  // Break out of the loop
      }

      offset++;                          // Increment offset into array
      if (offset >= BUFSIZE) offset = 0; // If the incoming data string is longer than our buffer, wrap around to avoid going out-of-bounds
    }
  }

  Serial.println(rfidData);       // The rfidData string should now contain the tag's unique ID with a null termination, so display it on the Serial Monitor
  Serial.flush();
  if(isThere){
  RFIDcounter++;
  Serial.print(RFIDcounter);
  isThere = false;
  }
```

Code

Beginning of loop - similar to line following, additions include adding a counter for the number of hashes it's reached, and then wrapping entire line following logic with if statement saying that it hasn't hit 5 dashes yet. Added while loop for communication into logic for when bot stops. Also added boolean variables to help with logic.

```
    if(HashCounter < 5){
    goForwardHash();   //call go foward hash function
    delay(500);     //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();
    }

  }

  else{

    //communicate
    if(shouldPrint == true){
    if(RFIDcounter == 0) Serial2.write('a');       //Send letter "s" out
    if(RFIDcounter == 1) Serial2.write('b');       //Send letter "s" out
    if(RFIDcounter == 2) Serial2.write('c');       //Send letter "s" out
    if(RFIDcounter == 3) Serial2.write('d');       //Send letter "s" out
    if(RFIDcounter == 4) Serial2.write('e');       //Send letter "s" out
    if(RFIDcounter == 5) Serial2.write('f');       //Send letter "s" out
    shouldPrint = false;
    servoStop();

    }
    }

  }
```

```
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

- Same as line following code.

```
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

- Same as line following code

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
- 4x Angle Brackets = $1.00

Total Estimated Cost To Date: $282.81

Updated 11/13/19 - Added 4 more angle brackets used to position sensor, total updated.

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%	100%
	Data Storage					50%
	Processing Data					50%
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%