New York City College of Technology

Dept. of Computer Engineering Technology

Class: EMT 2461 EM. Sys: Software Interface

Section: E356/Fall 2013

Instructor. Dr. Li

The Smart Express Bus

By Yeraldin Estrella, and Lulcasz Golbiewsk.

12/23/2013

Table of Contents

Project Description

Our project is a preliminary design for an automated bus. What that means is the bus supposed to detect different trigger points and perform different tasks, depending on the point detected.

For the project we have utilized two different colors of tape, yellow and black. Both tapes were 2 inches wide to make sure the robot has no problems with detection. The black tape has been used for the main track which the robot follows and as the initial trigger after Bluetooth initialization. The black tape was laid out for the maximum length of 5ft and 6 inches in a straight line. Within the maximum length of black tape we have set up two trigger points identified by yellow tape, which were 2 inches by 2 inches. The trigger points were spaced 16 inches apart from each other on the track. The first trigger point was set 30 inches from the beginning of the track.

We also utilized Infra-red sensors (IR Sensor) in our project. Front mounted IR sensor was used to detect an obstacle in the road and stop the robot until the obstacle was gone. The distance for the sensor was set to 440mm. if an object was pinged and returned the value of 440mm or less the robot stopped and held its position.

The side mounted IR was utilized for passenger detection. When the bus stop was trigged by yellow tape, the robot would ping the side and see if anybody was waiting at the bus stop. We used a block of wood to emulate people on the bus stop for the project.

If the condition came back negative for passengers, then we had the robot stop at the bus stop only for 1 second, but if there were passengers present then the time interval was 4 seconds before moving forward.

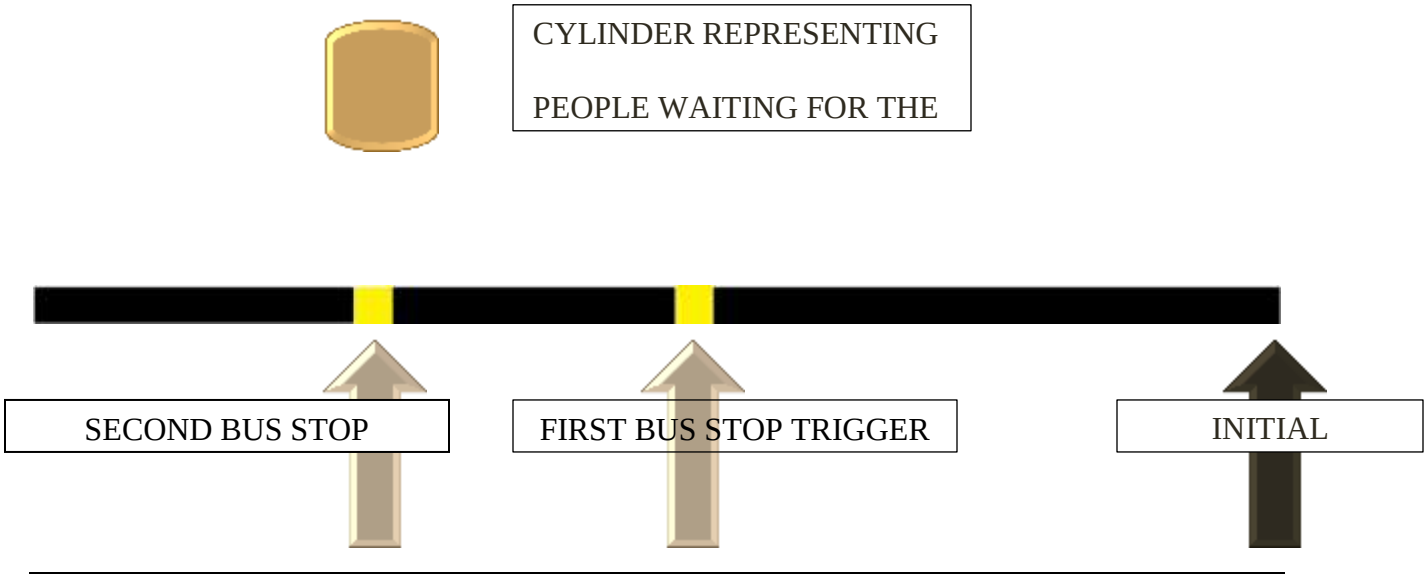
TRACK SET UP:

For the purposes of our project we have decided to have a track made up of two colors of 2 inch wide tape. The main color is black tape which represents main Street. The tape was put down on the floor, at the maximum length of 5ft and 6 inches, which is 66 inches. The yellow tape was placed 30 inches and 46 inches from the start location (16 inches apart).

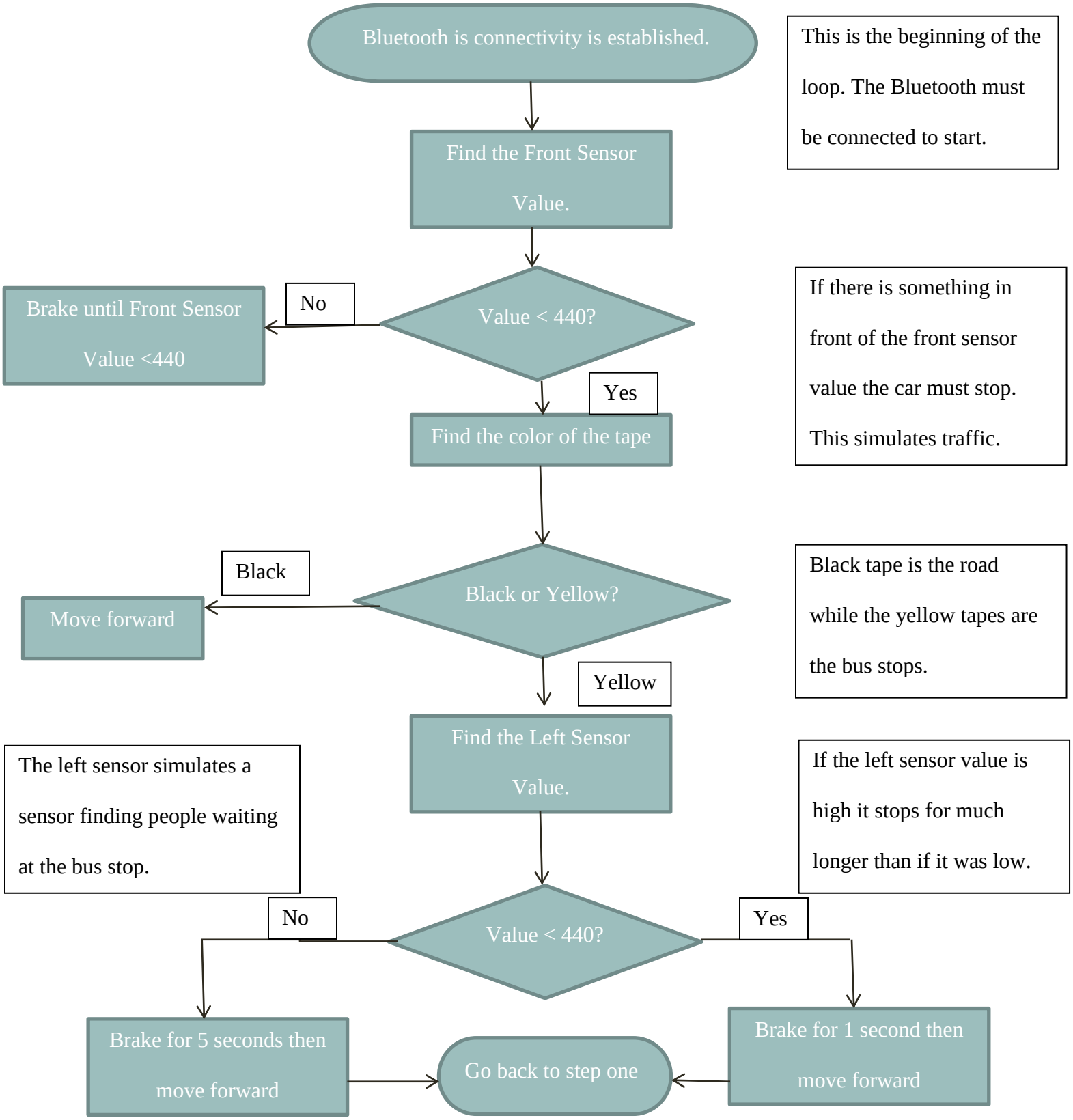
The initial position was represented by black tape, and for the robot to initialize we needed to the black tape detection as well as Bluetooth initialization via remote trigger. The necessary conditions under which the robot performs are:

- Color detection within acceptable range. (Range limit found using sensor calibration Black- between 200 and 400, Yellow-900 and 20000 [high variable fluctuation due to floor irregularity]).
- Movable obstacle.

TRACK SKETCH:



System Design



This is the beginning of the loop. The Bluetooth must be connected to start.

If there is something in front of the front sensor value the car must stop. This simulates traffic.

Black tape is the road while the yellow tapes are the bus stops.

The left sensor simulates a sensor finding people waiting at the bus stop.

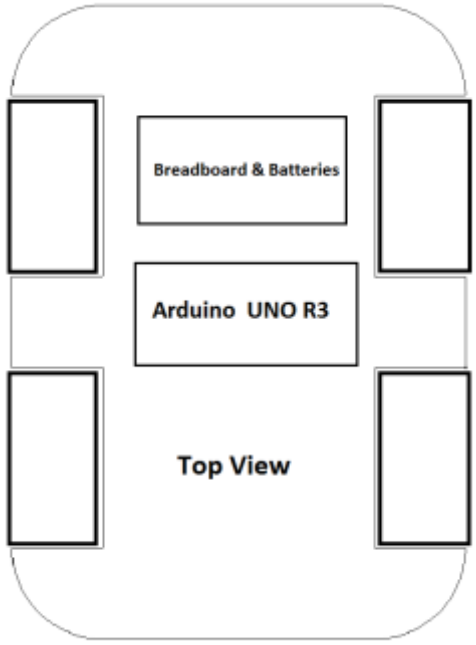
If the left sensor value is high it stops for much longer than if it was low.

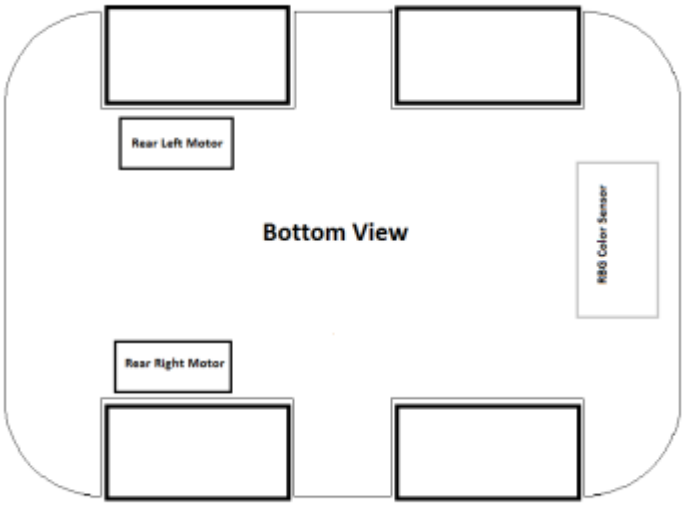
Component Design

Robot Layout

The following Drawings represent the Smart Express Bus layout and its mechanical and electrical Components.

Top and Bottom Views





Drawn by: Yeraldina Estrella

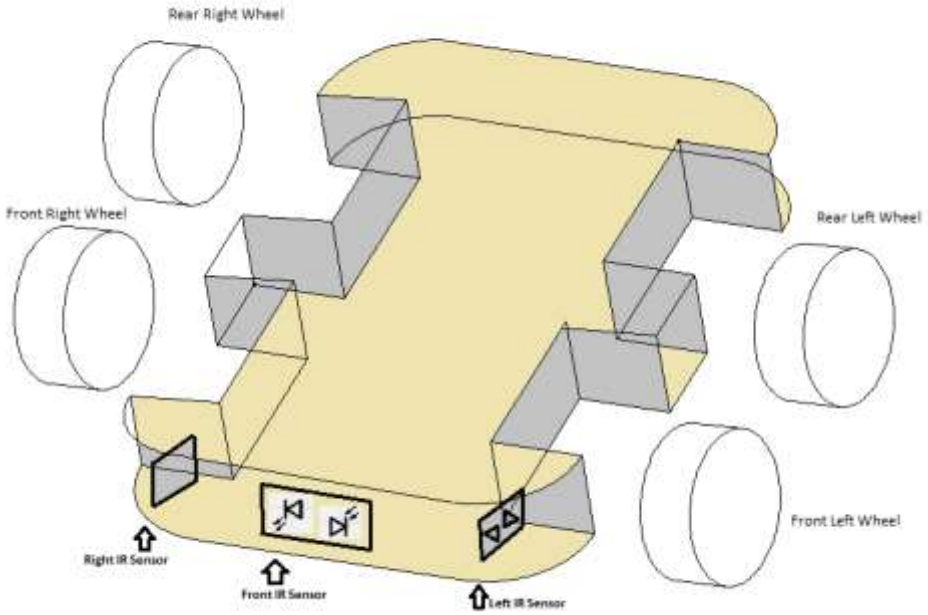
The Smart Express Bus was constructed using a chassis which is made of plastic and has four wheels. We mounted Arduino UNO R3 on the top as well as AA batteries to power Arduino and a breadboard for electrical components wiring and Bluetooth. The Bluetooth was used to allow communication from the smart express Bus and a smartphone.

For the purpose of this project, we use two DC brushed motors to drive the rear wheels (left and right) and the front wheels where utilized as freewheels meaning that we did not use motors to drive the front wheels.

We utilized an H-Bridge IC (SN754410) to drive the Rear Motors. The H-Bridge was mounted and wired using the breadboard on top of the chassis as shown in the Top View drawing. An RBG Color Sensor (TCS 34725) was mounted on the bottom of the chassis and was used to detect the line color in which the Smart Express Bus runs in this case black, as well as to detect the Bus stops which was represented by a different color(yellow).

We also utilized three Infrared Sensors (IR Sensors). One IR sensor was mounted on the front and the other two IR sensors were mounted on the right and left side as shown in the 3D drawing. The front IR sensor was utilized to detect traffic and avoid crashing with other objects ahead. The IR sensors on the sides were utilized to detect if passengers were waiting on the bus stop.

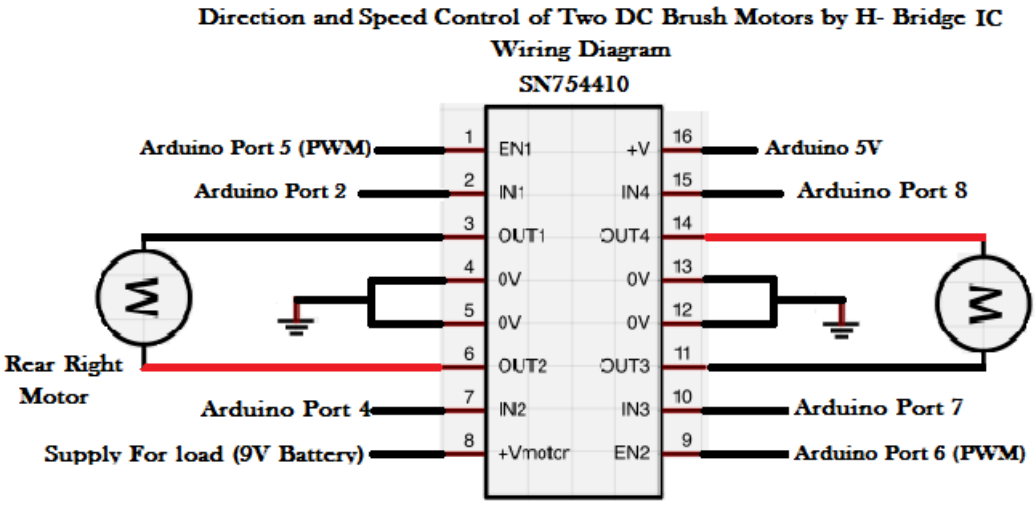
Robot 3D Drawing



Drawn by: Yeraldina Estrella

Wiring & Circuitry Schematics

H-Bridge IC SN754410 Wiring Diagram



Drawn by: Yeraldina Estrella

H-Bridge ICs allows controlling motors speed and direction. In order to control motor speed, the enable pin must be connected to a PWM Port in Arduino. PWM ports allow changing duty cycle and thus speed can be controlled. The motors need to connect to the output pins in each channel. The inputs pins needs to be connected to a port in Arduino. Pin 8 is the supply voltage for the motors and pin 16 is the supply voltage for the H-bridge usually connected to 5V from Arduino. Pins 4, 5, 12, and 13 are ground pins and must be connected to ground. The Arduino ports connected to the input pins enable 1 and enable 2 on the H-bridge must set as output.

For example:

```
int Enable1=5; //Enable1 connected to PWM port 5 on arduino
int Enable2=6;
pinMode(Enable1, OUTPUT);
```

```
pinMode(Enable2, OUTPUT);
pinMode(2, OUTPUT); //set arduino port 2 as output. Port 2 is connected to input 1 on H bridge
pinMode(4, OUTPUT); //set arduino port 4 as output. Port 7 is connected to input 2
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
```

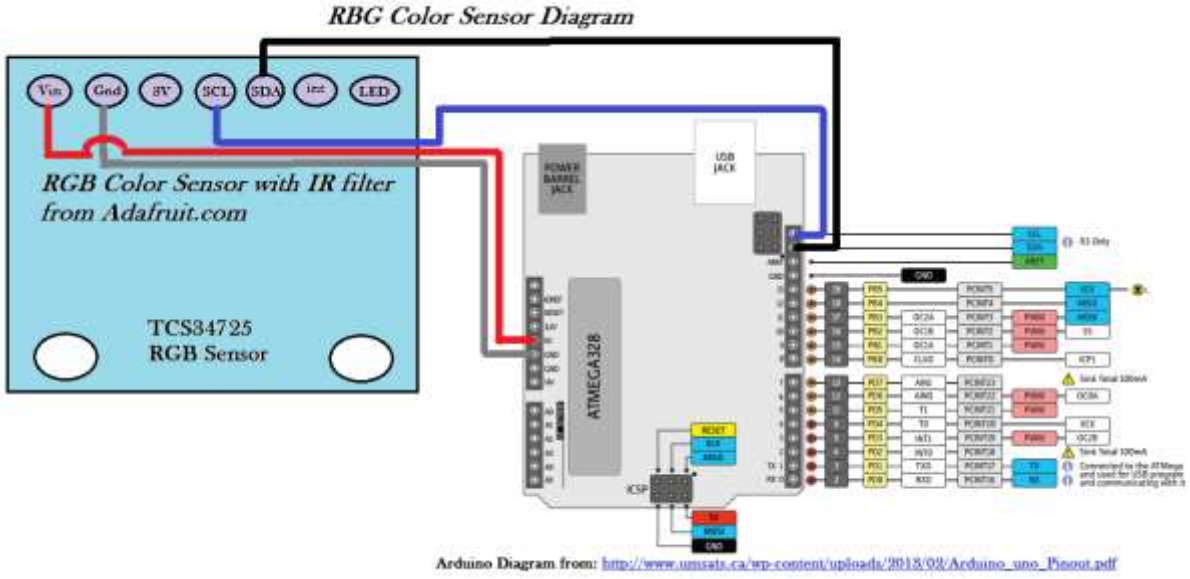
The motors speed can be controlled by changing the duty cycle. The direction in which the motors run can be control by changing the state of the pin that the motors are connected such as HIGH or LOW. The following code shows 100% duty cycle and the motors run forward:

```
 digitalWrite(2, LOW); // motors turn in FORWARD direction
digitalWrite(4, HIGH);
analogWrite(Enable1, 200); // full speed
digitalWrite(7, LOW); // motors turn in FORWARD direction
digitalWrite(8, HIGH);
analogWrite(Enable2, 200); // full speed
delay(2000);
```

The following table shows how the motor direction can be control.

Enable	Input 1	Input 2	Result
HIGH	HIGH	LOW	Motors turn Reverse
HIGH	HIGH	HIGH	Motors break
HIGH	LOW	HIGH	Motors Turn Forward
HIGH	LOW	LOW	Motors Break
LOW	X	X	Motors Break

Arduino UNO R3 & RBG Color Sensor Wiring Diagram



Arduino Uno R3 Diagram Source: http://www.umsats.ca/wp-content/uploads/2013/02/Arduino_uno_Pinout.pdf

Drawn by: Yeraldina Estrella

Arduino UNO R3 & RBG Color Sensor (TCS 34725) Wiring Table

RBG Color Sensor	Arduino
Vin	5V
Ground	Ground
SDL (Serial Data Line)	SDL (Serial Data Line)
SCL (Serial Clock Line)	SCL (Serial Clock Line)

For coding the RBG Color Sensor, the RBG Color Sensor library provided by Adafruit was used as follow in the sckecth:

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"

// our RGB -> eye-recognized gamma color
byte gammatable[256];
```

```
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
```

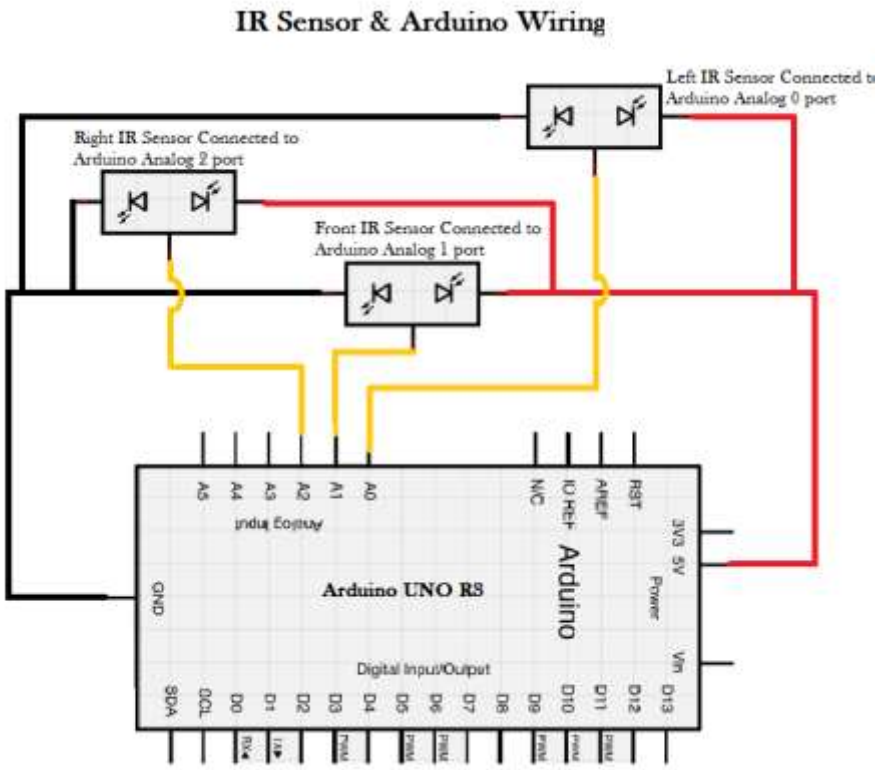
The following code was used to obtain color reading from the RBG color sensor:

```
1.  if (tcs.begin()) {
2.  Serial.println("Found sensor");
3.  } else {
4.  Serial.println("No TCS34725 found ... check your connections");
5.  while (1); // halt!
6.  }
7.  // it helps convert RGB colors to what humans see
8.  for (int i=0; i<256; i++) {
9.  float x = i;
10. x /= 255;
11. x = pow(x, 2.5);
12. x *= 255;
13. }
14. }
15. void loop() {
16. uint16_t clear, red, green, blue;
17. tcs.setInterrupt(false);      // turn on LED
18. delay(100);  // takes 50ms to read
19. tcs.getRawData(&red, &green, &blue, &clear);
20. tcs.setInterrupt(true);  // turn off LED
21. Serial.print("C:\t"); Serial.print(clear);
22. Serial.print("\tR:\t"); Serial.print(red);
23. Serial.print("\tG:\t"); Serial.print(green);
24. Serial.print("\tB:\t"); Serial.print(blue);



25. if ((clear >=200 && clear <= 400)) //&& (red >=110 && red <= 160)&& (green >=50 && green <= 110) && (blue >=30 &&
    blue <= 90))
26. {
27. Serial.println("black \n");
28. }
29. else if ((clear >=1000 && clear <= 1500))// && (red >=100 && red <= 400)&& (green >=100 && green <= 400) && (blue
    >=100 && blue <= 400))
30. {
31. Serial.println("yellow \n");
32. }
33. else if ((clear>=401 && clear<=600))
34. {
```

IR Sensor and Arduino Wiring



Drawn by: Yeraldina Estrella

The IR sensors are connected to ground and 5V and the data line is connected to an analog port in Arduino. The IR sensors works by transmitting light and receiving the light which is read as voltage depending on the wavelength of the light received. The IR sensor reading can be obtained by the following code:

```
1. int LeftSensorValue;
2. int RightSensorValue;
3. int FrontSensorValue;
4. void loop() {
5. // put your main code here, to run repeatedly:
6. LeftSensorValue=analogRead(0);
7. RightSensorValue=analogRead(2);
8. FrontSensorValue=analogRead(1);
```

Bluetooth Diagram

Drawn by: Yeraldina Estrella

Arduino UNO R3 & Bluetooth Wiring Table

Bluetooth	Arduino
5V	5V
Ground	Ground
TXD	RXD (Digital pin 0)
RXD	TXD (Digital pin 1)

Bluetooth can be used to communicate with Arduino. Android Smartphones allows communicating with a Bluetooth connected to Arduino by using an application such as Connection Terminal. Communication between Android phones and Arduino via a Bluetooth allows adding control to our smart express bus.

Arduino Sketch

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"
#include <SoftwareSerial.h>
SoftwareSerial myBluetooth(0,1);
int Enable1=5; //Enable1 connected to PWM port 5 on arduino
int Enable2=6;
void moveforward ();
void turnleft ();
void turnright();
void brake ();
// our RGB -> eye-recognized gamma color
byte gammatable[256];


Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

void setup() {
Serial.begin(9600);
pinMode(Enable1, OUTPUT);
pinMode(Enable2, OUTPUT);
pinMode(2, OUTPUT); //set arduino port 2 as output. Port 2 is connected to input 1 on H bridge
pinMode(4, OUTPUT); //set arduino port 4 as output. Port 7 is connceted to input 2
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);

pinMode(0, INPUT);
pinMode(1, OUTPUT);



  if (tcs.begin()) {
    Serial.println("Found sensor");
  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1); // halt!
  }

  // it helps convert RGB colors to what humans see
  for (int i=0; i<256; i++) {
    float x = i;
    x /= 255;
    x = pow(x, 2.5);
    x *= 255;

  }
 Serial.println("The Bus has left the station");
}

int LeftSensorValue;
int RightSensorValue;
int FrontSensorValue;

void loop() {
  uint16_t clear, red, green, blue;
  tcs.setInterrupt(false);      // turn on LED
  delay(50);  // takes 50ms to read
  tcs.getRawData(&red, &green, &blue, &clear);
  tcs.setInterrupt(true);  // turn off LED
   LeftSensorValue=analogRead(0);
```

```
  RightSensorValue=analogRead(2);
  FrontSensorValue=analogRead(1);


 if (myBluetooth.available())
{
 if (FrontSensorValue<=440)
 {

  if ((clear >=900 && clear <= 20000))// && (red >=100 && red <= 400)&& (green >=100 && green <= 400) && (blue
>=100 && blue <= 400))  //yellow
  {
     Serial.print("There are people waiting");

    if (RightSensorValue >= 440)
     {
        brake();
        delay (4000);
        moveforward;
        delay (300);

     }

     else if (RightSensorValue<=440)
     {
       brake();
       delay (1000);
       moveforward ();
       delay (300);
       Serial.print("There are people getting off");

    }

  }


  else if  (clear >=200 && clear <= 400) //&& (red >=110 && red <= 160)&& (green >=50 && green <= 110) && (blue
>=30 && blue <= 90))  //its black
  moveforward();

 }

   else if (FrontSensorValue>440)
   brake();
// }
//   else if((clear >=900 && clear <= 20000) && a==3)
//   Serial.println("This is the last stop");
}
void moveforward ()
{
   digitalWrite(2, LOW); // motors turn in FORWARD direction
   digitalWrite(4, HIGH);
   analogWrite(Enable1, 180);
   digitalWrite(7, LOW); // motors turn in FORWARD direction
   digitalWrite(8, HIGH);
   analogWrite(Enable2, 180);
   //delay (100);
}


void turnleft ()
{
```

```
   digitalWrite(2, HIGH); // motors turn in REVERSE direction
   digitalWrite(4, LOW);
   analogWrite(Enable1, 225); // full speed
   digitalWrite(7, LOW); // motors turn in FORWARD direction
   digitalWrite(8, HIGH);
   analogWrite(Enable2, 225); // full speed

}


void turnright ()
{
   digitalWrite(2, LOW); // motors turn in FORWARD direction
   digitalWrite(4, HIGH);
   analogWrite(Enable1, 0); //
   digitalWrite(7, HIGH); // motors turn in REVERSE direction
   digitalWrite(8, LOW);
   analogWrite(Enable2, 255); // full speed
   //delay(2500);
}

void brake ()
{
   digitalWrite(2, LOW); // motor brakes
   digitalWrite(4, LOW);
   analogWrite(Enable1, 220);
   digitalWrite(7, LOW); // motor brakes
   digitalWrite(8, LOW);
   analogWrite(Enable2, 220);

}


void reverse ()
{
   digitalWrite(2, HIGH); // motors turn in REVERSE direction
   digitalWrite(4, LOW);
   analogWrite(Enable1, 220);
   digitalWrite(7, HIGH); // motors turn in REVERSE direction
   digitalWrite(8, LOW);
   analogWrite(Enable2, 220);
   delay(250);
}
```

References

Predko, Michael. 123 PIC Microcontroller Experiments for the Evil Genius. New York: McGraw-Hill, 2005. Print.

Predko, Michael. 123 Robotics Experiments for the Evil Genius. New York: McGraw-Hill, 2004.Print.

Oxer, Jonathan, and Hugh Blemings. Practical Arduino: Cool Projects for Open Source Hardware. [Berkeley, CA]: Apress, 2009. Print.

Banzi, Massimo. Getting Started with Arduino. Beijing: O'Reilly, 2011. Print.

Evans, Brian. Beginning Arduino Programming. [New York]: Apress, 2011. Print.

Wheat, Dale. Arduino Internals. [United States]: Apress, 2011. Print.

Karvinen, Kimmo, and Tero Karvinen. Make Arduino Bots and Gadgets: Learning by Discovery. Beijing: O'Reilly, 2011. Print.

Monk, Simon. Arduino + Android Projects for the Evil Genius: Control Arduino with Your Smartphone or Tablet. New York: McGraw-Hill, 2012. Print.

Acknowledgement

This project was done by Lulcasz Golbiewsk and Yeraldina Estrella. Christopher Rivera built the chassis used for the project. Lulcasz Golbiewsk worked on the color sensor. Lulcasz provided the code for reading colors. Yeraldina Estrella did the wiring diagram for both the IR sensors and the H-bridge. Yeraldina wrote the code for controlling motor's speed and directions as well as IR sensor reading range for objects to be about 6 inches away from the IR sensor.