

**Department of Mechanical and Materials Engineering**

**Intelligent and Automation Laboratory**

**Tracking of Wall Following Robots using RFID Technology in Conjunction with Traffic Management**

Ben Church

# Summary

* The positions of two LYNXBots were interpolated between the four RFID readers placed, one on each side, around an approximately square testing arena.
* Information about the position of the robots was used to control traffic lights which dictate whether a robot was allowed to cross a bridge in the middle of the arena (see Figure 2).
* This was accomplished by using a MATLAB program which, upon receiving data from the Arduino board driving particular RFID reader, would run an animation showing the robot moving from the reader at which it was detected to the next reader and turn lights on and off based on where the robots were known to be.

# Background

# The idea of introducing RFID technology into MECH 452 began in 2013, with a study that examined the application of the technology to an automated warehouse task [1]. Work continued as a MECH 461 project in the winter of 2014 [2], with tests conducted to determine the optimum placement of the tags and the RFID reader.   This report documents the work of a 2014 summer project that set out to demonstrate the feasibility of a traffic management task, where 4 RFID readers were mounted under 4 speed bumps, placed behind 4 sets of traffic lights, and used to  track and control the location of mobile robots as they navigated an arena that mimicked a city streetscape.

# The LYNXBot

* The LYNXBots use SHARP infrared distance sensors to follow the outside wall of the testing arena in the counter-clockwise direction.
* The LYNXBots were programmed to maintain a particular distance, as measured by the SHARP, from the wall of the arena. This allows them to turn the corners and ensures that they drive over the RFID readers. See Figure 1 for RFID card placement under the robots.
* The robots were programmed to count corners so that once on the bridge, they would drive only straight until they were back on the outside of the arena.
* The robots also had forward facing SHARP sensors to prevent collisions should the traffic light system fail.



Figure 1: The underside of one of the LYNXBots showing RFID card placement.

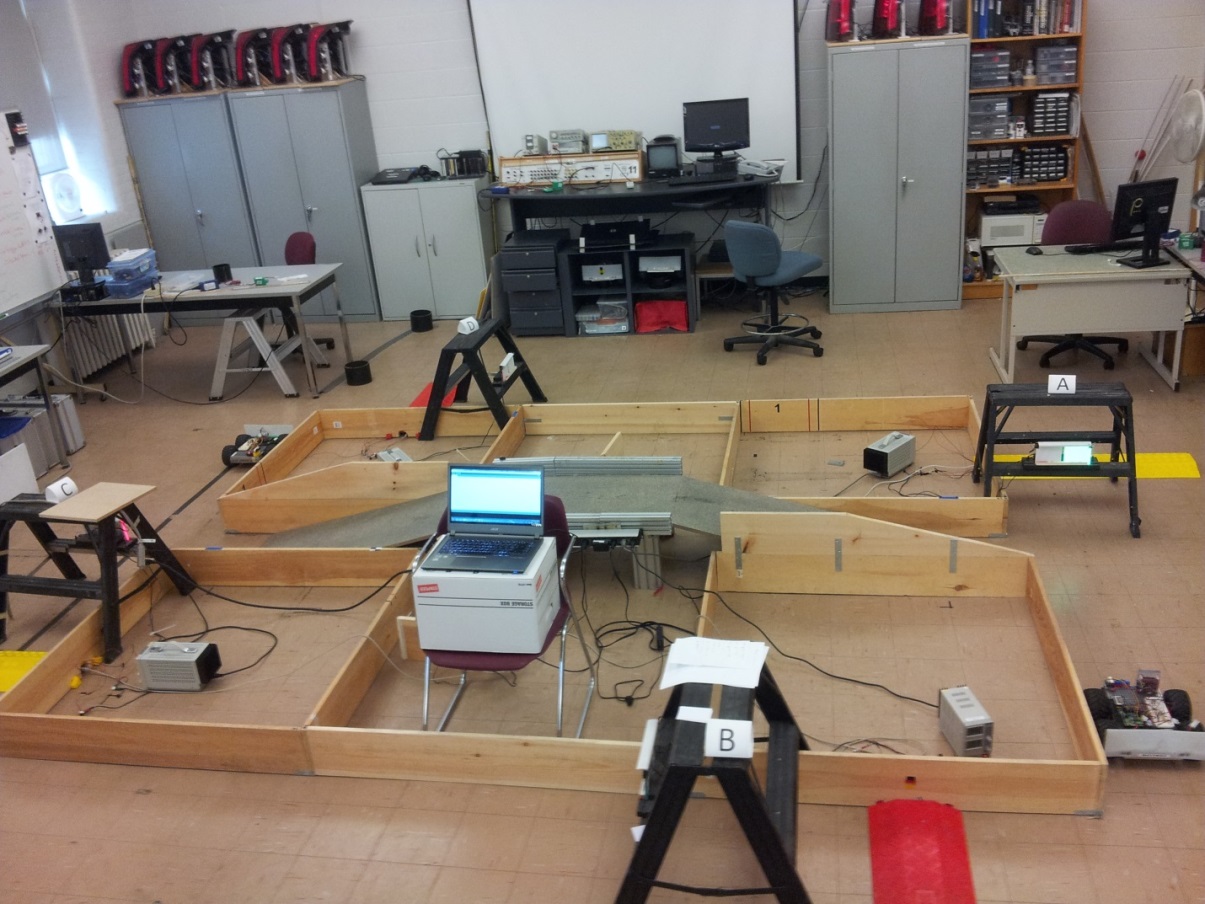


Figure 2: Photograph showing the testing arena set-up.

# The Arduino/Traffic Light Driver Circuits

* Adafruit PN532 RFID readers were integrated into an Arduino circuit and mounted underneath speed bumps so that the LYNXBots could drive over them and have the RFID tags on their undersides’ detected. See Figure 3 for reader placement under the speed bump.
* The Adafruit\_PN532 library [3] was downloaded so that the Arduino program [3] provided by Adafruit Industries could be used
* Unnecessary code (such as that required to read other RFID tag types) was removed from the program for the sake of speed.
* ULN2003 chips supplied with 12V were used to pull the ground terminals of the LEDs to ground when supplied with a control signal indicating that the light was to be turned on.
* Figure 4 shows the circuit diagram for the RFID reader and light control.

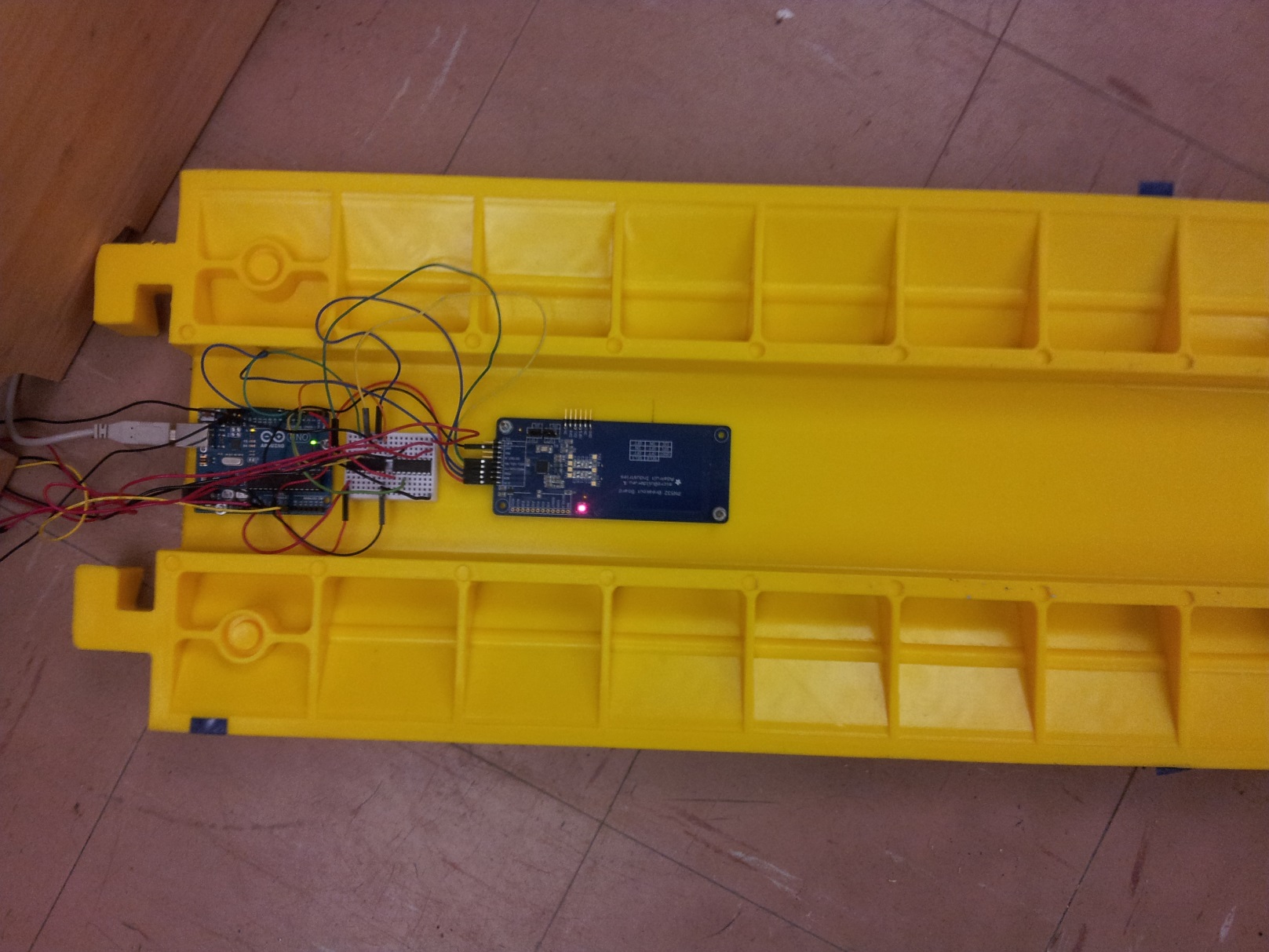
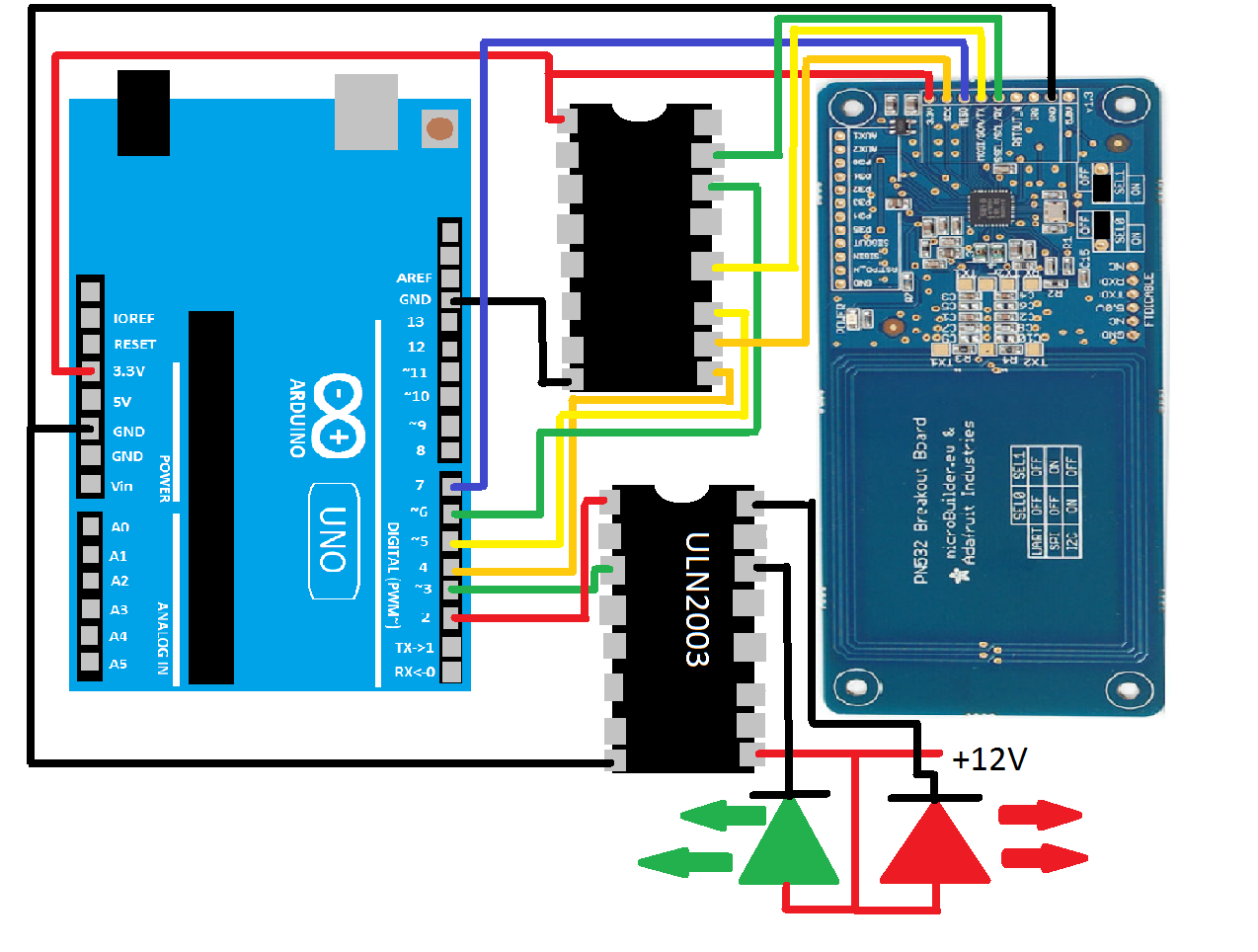


Figure 3: The underside of one of the speed bumps showing the RFID circuit arrangement.

RFID Reader and Traffic Light Driver Circuit Diagram



USB

Figure 4: Circuit schematic showing RFID reader assembly and ULN2003 used to control lights. PN532 image is from <https://learn.adafruit.com/assets/3271>. Note that some of the wires depicted here as orange may be blue in the actual circuit for lack of orange wires.

# The MATLAB Program with Robot Animation

* While the program is running, MATLAB reads data from the Ardunios’ buffers, if there is data corresponding to an RFID tag to be read.
* The program compares this data against a lookup table to determine which robot was detected at whichever reader sent the data.
* When a robot is detected, the program then animates a rectangle (patch in MATLAB) moving from one reader to the subsequent reader. This will look something like Figure 5.
* If the light ahead of the robot is red, the animated robot pauses as soon as it can ‘see’ the light and continues once the light turns green, as dictated by the other robot having cleared the bridge.
* Any robot being animated is moved through the animation by a certain increment (number of frames) with each loop of the program so that multiple robots can be plotted independently.
* If the animation runs to completion before the robot is detected at the next reader, the increment by which that robot will move in one loop through any animation is divided by the ratio of the time it took for the robot to be detected at the second reader after being detected at the first to the time that the animation took to run.
* If the robot is detected at the next reader before the animation is complete, the increment is multiplied by the ratio of the number of frames in the animation to the frame the animation reached when the robot was detected at the next reader.

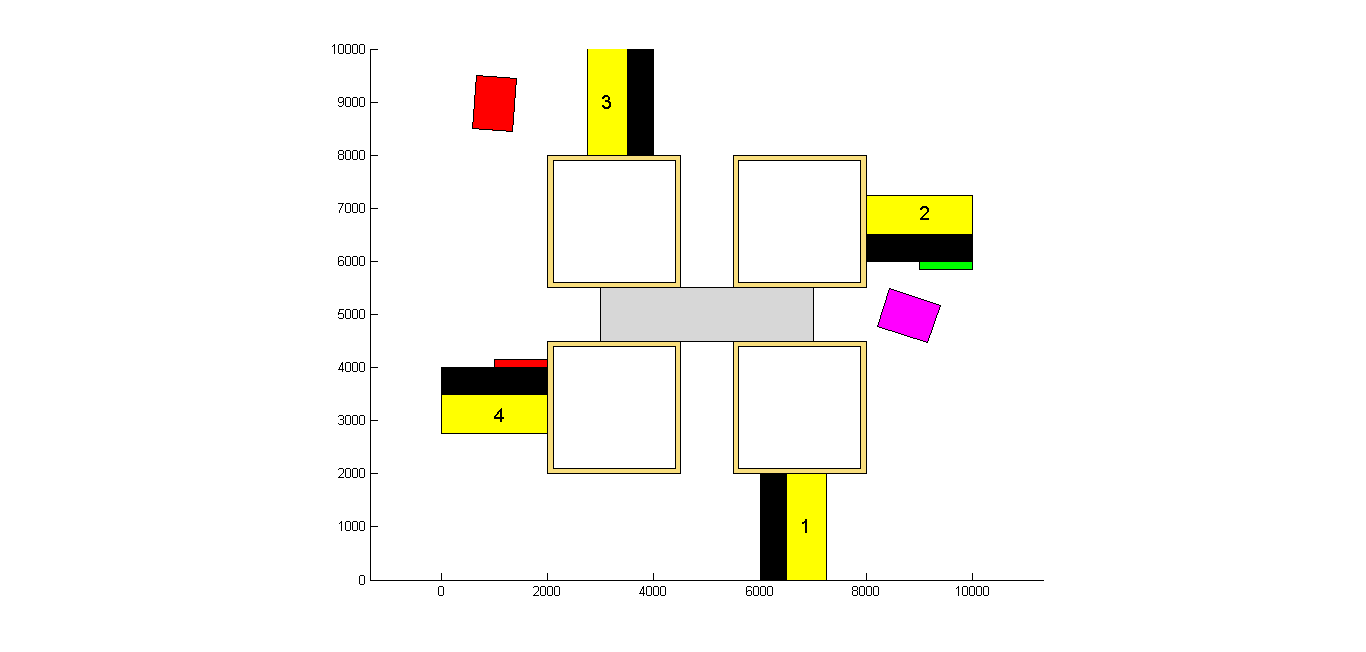


Figure 5: Snap shot of the figure MATLAB plots the robots’ motion on.

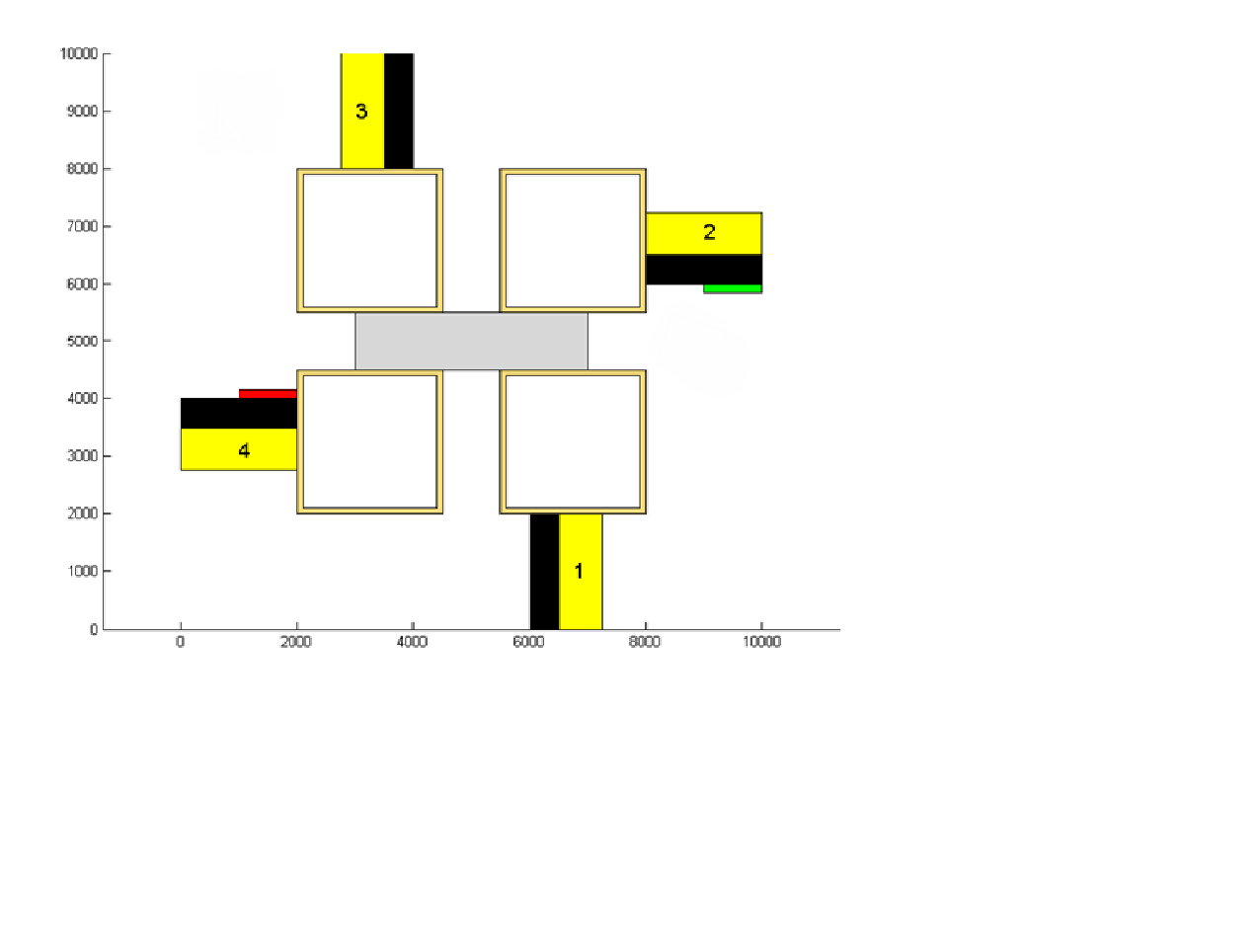


Figure 6: A graphical representation of each animation the main program can call.

* The animations are enumerated according to which RFID reader they begin at. These numbers are seen in their function names (‘red\_1’, green\_3’, etc…).
* Each animation begins with the robot being drawn at the reader that it is detected at and has it stop approximately at the head of its arrow. The green animations stop on the subsequent readers in anticipation of the robot being detected there, which would initiate the next animation.
* As the code is written (see the appendix), ‘red\_2’ and ‘red\_4’ will never occur since lights #1 and #3 are not being controlled, therefore, ‘green\_2’, and ‘green\_4’ run every time a robot is detected at readers #2 and #4, respectively.
* ‘red\_1’ and ‘red\_3’ occur when lights #2 and #4 are red, respectively. Similarly for ‘green\_1’, and ‘green\_3’ with their corresponding green lights.
* If the following light changes from red to green during or after the red animation, the green animation will being at the end of the red animation.
* The speed at which each animation progresses is controlled by changing the number of frames the robot will be moved forward in a single iteration. The red animations are 4000 frames long and the green animations are 22000 frames long. The initial number of frames that a robot moves in one iteration is 100. This number of frames is assigned to the individual robots, rather than the animations, to correct for the robots’ speeds.
* Figure 6 shows a graphical representation of the animation system of the existing program. The animations behave as follows:
  1. Light #2 is green and light #4 is red.
  2. Robot #1 starts in the lower right corner of the arena, facing light #2 while robot #2 starts in the top left corner facing light #4.
  3. Robot #1, seeing the green light, will proceed across the bridge while robot #2 waits at its red light.
  4. Once robot #1 has crossed the bridge and is detected at reader #4, light #2 will change to red and light #4 will change to green.
  5. Robot #2, now looking at a green light will turn onto the bridge and eventually be detected at reader #2, toggling the lights again.
  6. Robot #1, after being detected at reader #4 will begin ‘green\_4’ in which the animated robot turns up the corridor. However, the robot will continue along the wall and be detected at reader #1, beginning ‘red\_1’ as the lights were changed when it was detected at reader #4.
  7. Robot #1 will stop when it sees that light #2 is red until robot #2 is detected at reader #2, which will change the lights back to their initial conditions, allowing the cycle to repeat.
  8. ‘green\_2’ will be run for robot #2 but, like for robot #1, this is corrected when the robot reaches the next reader (reader #3).

# The MATLAB Program only for Light Control and Display

* The animation components of the original program were removed, resulting in a program which controls the lights according to the desired logic.
* The program still draws a figure which shows the states of the lights in the arena, but does not animate the robots’ motion.

# Future Recommendations

* The animation could be modified in accordance with the sequence of operations depicted in Figure 7 and described below it.

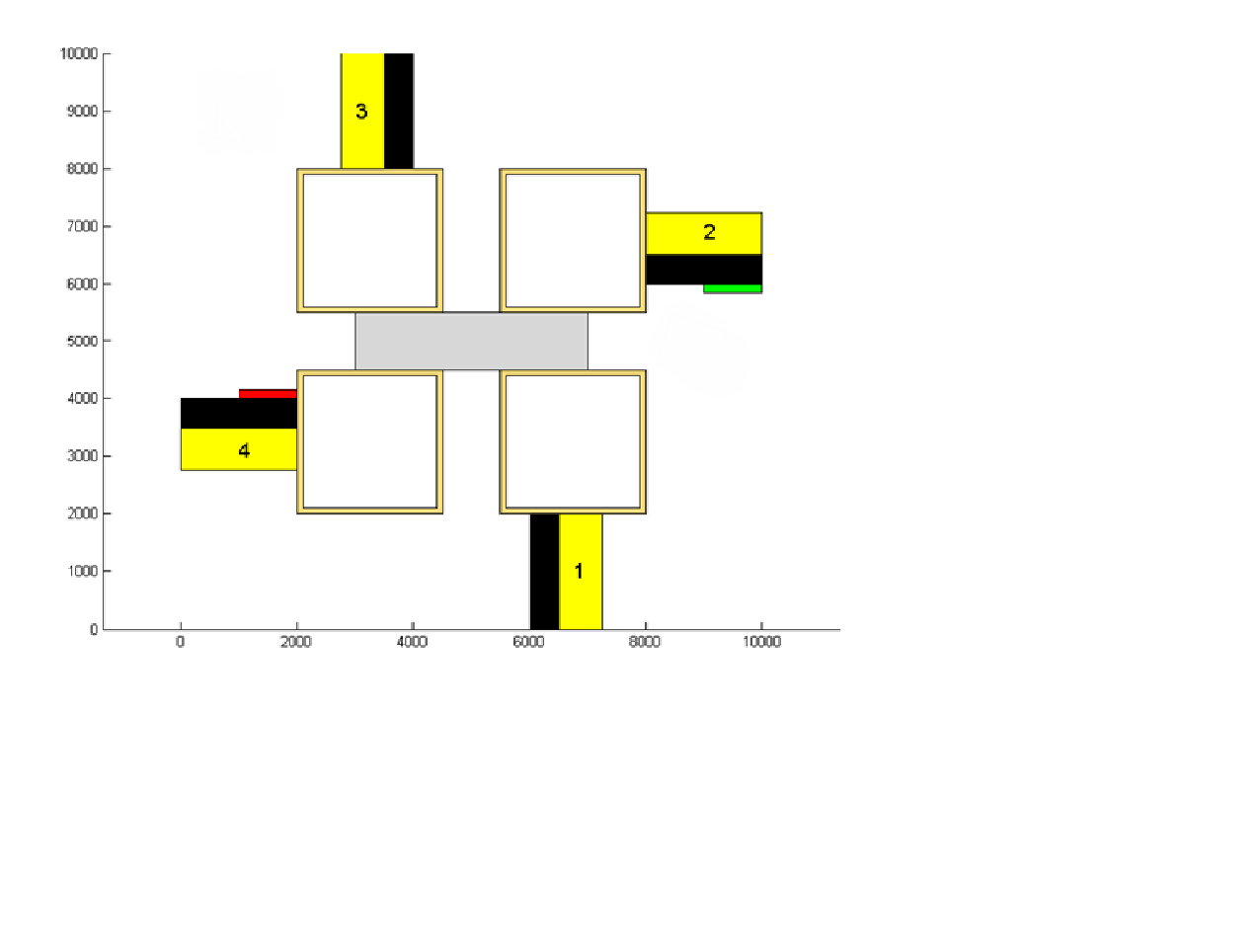


Figure 7: Proposed system of animations for future work.

* The above system of animations would behave as follows:
  + As in the original system of animations, robot #1 begins in the corner of the arena corresponding to the lower right corner of Figure 7 facing light #2, which is initially green. Robot #2 starts in the corner of the arena corresponding to the top left corner of Figure 7 facing light #4 which is initially red.
  + Robot #1, seeing a green light, will proceed along the path of the green arrow. The animation will keep going past reader #4 until the robot is detected there. A robot being detected at reader #4 will change light #4 from red to green and light #2 from green to red.
  + Robot #2 will now cross the bridge opposite to the direction that robot #1 just did.
  + Robot #1, having just been detected at reader #4 will follow the animation corresponding to the white arrow. This animation will continue on indefinitely past reader #1 until robot #1 is detected at reader #1.
  + As light #2 should still be red due to robot #1 being detected at reader #4, robot #1 will now follow the path corresponding to the red arrow leaving reader #1. The robot will stop at the end of this animation and wait for robot #2 to be detected at reader #2, which will change the lights back to their initial state, restarting the cycle.

# References

1. Fernando, F. and Surgenor, B. (2013) “An RFID-based Automated Warehouse Project for a Course in Mechatronics”, Proc. Int. Conf. on Mechanical Engineering and Mechatronics”, Toronto, Aug 8 to 9.
2. Lounsbury, B. (2014) “Traffic Management of Automated Guided Vehicles Using RFID Technology”, MECH 462 Report, Queens University, March
3. Adafruit, (2014) “Arduino library for SPI access to the PN532 RFID”, <https://github.com/adafruit/Adafruit-PN532>

# Appendix

This appendix lists the following six programs:

* half\_arena\_camera\_bot.ino
  + This program was uploaded onto BOT#S9 and allows it to navigate the arena by means of a wall following algorithm. The robot counts corners so that it will only drive straight once it is on the bridge. This program also implements a forward facing SHARP sensor which was used to stop the robot by blocking its path since a camera had not been implemented to work with the lights.
* RFID\_read.ino
  + This is the program used by Arduino to control the RFID reader as well as the traffic lights.
* full\_arena\_two\_bots.m
  + This is the main program run in MATLAB which keeps track of the positions of the robots, calls the functions which animate the robots, and sends the appropriate data to the Arduino circuits to control the traffic lights.
* red\_1.m
  + This is the animation that occurs when a robot is detected at reader #1 as shown in Figure 6, if light #2 is red.
* green\_1.m
  + This is the animation that occurs when a robot is detected at reader #1 as shown in Figure 6 and light #2 is green or if light #2 changes from red to green after ‘red\_1.m’ has finished.
* task\_two\_just\_lights.m
  + This is the version of ‘full\_arena\_two\_bots.m’ simplified so as to no longer plot the positions of the robots but only to control and display the states of the traffic lights.

Sample LYNXBot Code (used on BOT#S9)

/\*

Heshan\_wallFollowSquareProp

Created by: Heshan Fernando

Date: 09/24/2013

Proportional solution for Lab 5 Part 3 using Bot A1

\*/

// Pin Assignments

int RED = 4;

int YLW = 5;

int GRN = 6;

int BUTTON\_A = 7;

int BUTTON\_B = 8;

int BUTTON\_C = 9;

int MOTOR\_L = 10; // left motor PWM signal

int MOTOR\_R = 11; // right motor PWM signal

int BUMPER = 13;

int SHARP\_1 = A1; //sharp input pin

int SHARP\_2 = A2;

//global constants

const int STOP\_SPEED = 145;//pulse length for stopping motors

const int DELTA = 28; //forward speed.

const int TARGET = 1700; //sharp for target distance, 1400 is approximate

const float GAIN = 0.45;

//global variables

boolean wall\_lost = false;

boolean look\_for\_edge = true;

boolean four\_corners = false;

int corners = 0;

unsigned long start\_time;

// Set-UP

// the setup routine runs once when you press reset:

void setup() {

// initialize the digital led pins as outputs.

pinMode(RED, OUTPUT);

pinMode(YLW, OUTPUT);

pinMode(GRN, OUTPUT);

//initialize buttons and bumper pins as inputs

pinMode(BUMPER, INPUT);

pinMode(BUTTON\_A, INPUT);

pinMode(BUTTON\_B, INPUT);

pinMode(BUTTON\_C, INPUT);

//initialize motor control pins as outputs

pinMode(MOTOR\_L, OUTPUT);

pinMode(MOTOR\_R, OUTPUT);

//setup serial debug

Serial.begin(9600);

delay(200);

//Plot Lite Program Setup //carriage return

Serial.print("!USRS Sharp Test Plot"); Serial.write(13); // plot title

Serial.println("!SPAN 800, 3000"); Serial.write(13); // set vertical range

Serial.println("!TMIN 0"); Serial.write(13); // set minimum time

Serial.println("!TMAX 20"); Serial.write(13); // set maximum time

Serial.println("!PNTS 400"); Serial.write(13); // set number of points

Serial.println("!MAXS"); Serial.write(13); // set to stop when max reached

runMotors(0,0); //stop motors

Serial.println("Press Button A to start.");

do {

toggleLED(GRN); //motors stopped, Green LED flashing

} while(digitalRead(BUTTON\_A)== HIGH);

Serial.println("Program Running. Press bumper to stop");

}

// Main Program Loop

// the loop routine runs over and over again forever

void loop() {

int sensor = map(analogRead(SHARP\_1), 0, 1023, 0, 5000);

int foreward = map(analogRead(SHARP\_2),0,1023,0,5000);

if(foreward < 2000)

{

int error = TARGET - sensor;

int dummy = min(abs(error)\*GAIN, 100);

int delta\_v = DELTA\*(100-dummy)/100;

if((corners == 2) || (corners == 7)){

delta\_v = DELTA;

}

//comment out the following lines to speed up the program

//plotLite(sensor); //send value to plotlite

// Serial.print("error: ");

// Serial.print(error);

// Serial.print("\t");

// Serial.print("dummy: ");

// Serial.print(dummy);

// Serial.print("\t");

// Serial.print("delta\_v: ");

// Serial.print(delta\_v);

if(error > 0 ){ //too far from the wall

//turnOnLED(YLW);

runMotors(delta\_v, DELTA);

}else if(error < 0){ //too close to the wall

//turnOnLED(RED);

runMotors(DELTA, delta\_v);

}else{ //desired distance

//turnOnLED(GRN);

runMotors(DELTA,DELTA);

}

//if corner is detected

if(sensor < 900 && look\_for\_edge){

if(corners == 0 || corners == 5) {

corners++;

wall\_lost = true;

turnOnLED(YLW);

// runMotors(DELTA,DELTA);

// delay(400);

// runMotors(0,0);

// delay(3000);

// runMotors(-DELTA,DELTA);

// delay(600);

// runMotors(DELTA,DELTA);

// delay(400);

}

else if(corners == 3 || corners == 8) {

corners++;

wall\_lost = true;

turnOnLED(YLW);

// runMotors(DELTA,DELTA);

// delay(200);

// runMotors(-DELTA,DELTA);

// delay(700);

// runMotors(DELTA,DELTA);

// delay(1600);

// runMotors(0,0);

// delay(3000);

}

else {

corners++;

wall\_lost = true;

turnOnLED(YLW);

}

}

if(wall\_lost){

look\_for\_edge = false;

if(sensor > 1400){ //when wall is found again

look\_for\_edge = true;

wall\_lost = false; //reset the variable

Serial.println(corners);

turnOnLED(GRN);

}

}

if (corners == 10){

start\_time = millis();

four\_corners = true;

}

if (four\_corners){

if((millis() - start\_time) > 1400){

runMotors(0,0);

while(1){

flashAllLEDs();

}

}

}

if(digitalRead(BUMPER) == HIGH){

runMotors(0,0);

while(1){

flashAllLEDs();

}

}

}

else

{

runMotors(0,0);

delay(3000);

}

}

//\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Turn on a single LED, and all other off

void turnOnLED(int colour){

digitalWrite(GRN, LOW);

digitalWrite(YLW, LOW);

digitalWrite(RED, LOW);

digitalWrite(colour, HIGH);

}

//Toggle an LED on/off

void toggleLED(int colour){

digitalWrite(colour, HIGH);

delay(250);

digitalWrite(colour, LOW);

delay(250);

}

//flash all LEDs

void flashAllLEDs(){

digitalWrite(GRN, LOW);

digitalWrite(YLW, LOW);

digitalWrite(RED, LOW);

delay(250);

digitalWrite(GRN, HIGH);

digitalWrite(YLW, HIGH);

digitalWrite(RED, HIGH);

delay(250);

}

void runMotors(int delta\_L, int delta\_R)

{

int pulse\_L = (STOP\_SPEED + delta\_L)\*10; //determines the length of pulse in microseconds

int pulse\_R = (STOP\_SPEED + delta\_R)\*10;

for(int i=0; i<3; i++){

pulseOut(MOTOR\_L, pulse\_L); //send pulse to left motors

pulseOut(MOTOR\_R, pulse\_R); //send pulse to right motors

}

}

void pulseOut(int motor, int pulsewidth)

{

digitalWrite(motor, HIGH);

delayMicroseconds(pulsewidth); //send pulse of desired pulsewidth

digitalWrite(motor, LOW);

}

void plotLite(int value) {

Serial.print(value); //output to plotlite

Serial.write(13); //carriage return

delay(50);

}

Arduino RFID Reader and Light Control Program Listing

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@file readMifare.pde

@author Adafruit Industries

@license BSD (see license.txt)

This example will wait for any ISO14443A card or tag, and

depending on the size of the UID will attempt to read from it.

If the card has a 4-byte UID it is probably a Mifare

Classic card, and the following steps are taken:

- Authenticate block 4 (the first block of Sector 1) using

the default KEYA of 0XFF 0XFF 0XFF 0XFF 0XFF 0XFF

- If authentication succeeds, we can then read any of the

4 blocks in that sector (though only block 4 is read here)

If the card has a 7-byte UID it is probably a Mifare

Ultralight card, and the 4 byte pages can be read directly.

Page 4 is read by default since this is the first 'general-

purpose' page on the tags.

This is an example sketch for the Adafruit PN532 NFC/RFID breakout boards

This library works with the Adafruit NFC breakout

----> https://www.adafruit.com/products/364

Check out the links above for our tutorials and wiring diagrams

These chips use SPI to communicate, 4 required to interface

Adafruit invests time and resources providing this open source code,

please support Adafruit and open-source hardware by purchasing

products from Adafruit!

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <Adafruit\_PN532.h>

#define SCK (4)

#define MOSI (5)

#define SS (6)

#define MISO (7)

Adafruit\_PN532 nfc(SCK, MISO, MOSI, SS);

int red\_light = 2;

int green\_light = 3;

int lights = 0;

int sharp = 0;

int trip = 0;

void setup(void) {

Serial.begin(9600);

//Serial.println("Hello!");

nfc.begin();

uint32\_t versiondata = nfc.getFirmwareVersion();

if (! versiondata) {

Serial.print("Didn't find PN53x board");

while (1); // halt

}

nfc.SAMConfig();

pinMode(red\_light, OUTPUT);

pinMode(green\_light, OUTPUT);

}

void loop(void) {

//digitalWrite(red\_light, HIGH);

//digitalWrite(green\_light, LOW);

if(lights == 0)

{digitalWrite(red\_light, LOW);

digitalWrite(green\_light, LOW);}

else if(lights == 1)

{digitalWrite(red\_light, HIGH);

digitalWrite(green\_light, LOW);}

else if(lights == 2)

{digitalWrite(red\_light, LOW);

digitalWrite(green\_light, HIGH);}

else

{digitalWrite(red\_light, HIGH);

digitalWrite(green\_light, HIGH);}

trip = analogRead(sharp);

if(trip > 300)

{

Serial.println(trip);

}

uint8\_t success;

uint8\_t uid[] = { 0, 0, 0, 0, 0, 0, 0 }; // Buffer to store the returned UID

uint8\_t uidLength; // Length of the UID (4 or 7 bytes depending on ISO14443A card type)

success = nfc.readPassiveTargetID(PN532\_MIFARE\_ISO14443A, uid, &uidLength);

if (success) {

nfc.PrintHex(uid, uidLength);

}

if(Serial.available()>0)

{

//Serial.println(Serial.read());

lights = Serial.read();

}

}

Main MATLAB Program Listing for Robot Animation

clear all; clc; clf; close all;

%% Initialize plot/animation

figure(1)

hold all

xlim([0, 10000]);

ylim([0, 10000]);

axis equal

rectangle('position', [2000, 2000, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % Plot

rectangle('position', [2100, 2100, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [2000, 5500, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % all

rectangle('position', [2100, 5600, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [5500, 5500, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % the

rectangle('position', [5600, 5600, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [5500, 2000, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % boxes

rectangle('position', [5600, 2100, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [6000, 0, 500, 2000], 'facecolor','k')

r(1) = rectangle('position', [5850, 1000, 150, 1000], 'facecolor', 'r');

g(1) = rectangle('position', [5850, 0, 150, 1000], 'facecolor', 'g');

rectangle('position', [8000, 6000, 2000, 500], 'facecolor', 'k')

r(2) = rectangle('position', [8000, 5850, 1000, 150], 'facecolor', 'r');

g(2) = rectangle('position', [9000, 5850, 1000, 150], 'facecolor', 'g');

rectangle('position', [3500, 8000, 500, 2000], 'facecolor', 'k')

r(3) = rectangle('position', [4000, 8000, 150, 1000], 'facecolor', 'r');

g(3) = rectangle('position', [4000, 9000, 150, 1000], 'facecolor', 'g'); % Plot ladders and their lights

rectangle('position', [0, 3500, 2000, 500], 'facecolor', 'k')

r(4) = rectangle('position', [1000, 4000, 1000, 150], 'facecolor', 'r');

g(4) = rectangle('position', [0, 4000, 1000, 150], 'facecolor', 'g');

for z = 1:4 % All lights are initially off

set(g(z), 'visible', 'off')

set(r(z), 'visible', 'off')

end

rectangle('position', [6500, 0, 750, 2000], 'facecolor', 'y');

text(6760, 1000, '1', 'fontsize', 14)

rectangle('position', [8000, 6500, 2000, 750], 'facecolor', 'y');

text(9000, 6900, '2', 'fontsize', 14)

rectangle('position', [2750, 8000, 750, 2000], 'facecolor', 'y') % Plot and enumerate RFID speed bumps

text(3010, 9000, '3', 'fontsize', 14)

rectangle('position', [0, 2750, 2000, 750], 'facecolor', 'y')

text(1000, 3100, '4', 'fontsize', 14)

rectangle('position', [3000, 4500, 4000, 1000],...

'facecolor', [(215/255),(215/255),(215/255)]) % Plot bridge

p(1:4) = patch([0,0,0,0],[0,0,0,0],'k'); % Preallocate patches that will represent robots

red\_frame = zeros(1,4); % Initialize frame of red light animations

red\_increment(1:4) = 100\*ones(1,4); % How far the animation will proceed each step

green\_frame = zeros(1,4);

green\_increment(1:4) = 100\*ones(1,4);

t\_plot = zeros(1,4); % Initialize variables to be used to time animaions for speed correction

t\_restart = zeros(1,4);

red\_complete = zeros(1,4); % Logic values representing whether an animation is finished or not

green\_complete = zeros(1,4);

green\_start = zeros(1,4);

%% Initialize RFID lookup table and RFID readers

delete(instrfind); % Close any serial device communications left open

id\_or\_sharp = {0,0,0,0}; % Create variable in which to store data from Arduino circuit

rfid = zeros(4);

fields = {'robot1', 'robot2', 'robot3', 'robot4'};

robots = {'Robot #1', 'Robot #2', 'Robot #3', 'Robot #4'};

readers = {'Reader #1', 'Reader #2', 'Reader #3', 'Reader #4'};

reader(1) = serial('COM8','BaudRate',9600);

reader(2) = serial('COM7','BaudRate',9600); % Connect to the COM ports

reader(3) = serial('COM6','BaudRate',9600);

reader(4) = serial('COM9','BaudRate',9600);

for i = 1:4;

fopen(reader(i)); % Open communication on the COM ports

flushinput(reader(i)); % Clear Arduinos' serial buffers

end

rfid\_table = struct('robot1', {'0xFD 0x8C 0x82 0x9A','0x95 0x6C 0x59 0x6F'},...

'robot2', {'0x7D 0xFE 0x88 0x9A','0x5D 0xEE 0x82 0x9A'},...

'robot3', {'0x25 0x68 0xA2 0x51','0x35 0xE9 0x7D 0x33'},... % Robot lookup table

'robot4', {'0xED 0x1E 0x86 0x9A','0x2D 0x5A 0x83 0x9A'});

lights = zeros(4,1); % The state of each light set is represented by one element of the variable

isred = zeros(1,4);

q = 1;

while q < 5; % For some reason, this loop allows lights to be turned on...

fscanf(reader(q)); % ...(initial conditions) without an RFID tag yet being scanned.

lights(2) = 2;

set(g(2),'visible','on')

set(r(2),'visible','off')

fwrite(reader(2),lights(2))

lights(4) = 1;

set(g(4),'visible','off')

set(r(4),'visible','on')

fwrite(reader(4),lights(4))

isred(4) = 1;

q = q+1;

end

%% Scan for robots and plot them

while 1

for i = 1:4

if reader(i).BytesAvailable % If there is anything in circuits' buffers

id\_or\_sharp(i) = {fscanf(reader(i))}; % Store buffer data (SHARP or PN532) in a variable

flushinput(reader(i)) % Clear buffer to prevent unnecessary entry into loops

if length(id\_or\_sharp{i}) == 21 || (length(id\_or\_sharp{i}) == 1 && id\_or\_sharp{i} == 1); % If data is an RFID tag ID or the loop should be repeated

if i == 2 % Should the robot cross the bridge?

lights(2) = 2;

set(g(2),'visible','on');

set(r(2),'visible','off');

fwrite(reader(2),lights(2));

isred(2) = 0;

lights(4) = 1;

set(g(4),'visible','off');

set(r(4),'visible','on');

fwrite(reader(4),lights(4));

isred(4) = 1;

elseif i == 4

lights(2) = 1;

set(g(2),'visible','off');

set(r(2),'visible','on');

fwrite(reader(2),lights(2));

isred(2) = 1;

lights(4) = 2;

set(g(4),'visible','on');

set(r(4),'visible','off');

fwrite(reader(4),lights(4));

isred(4) = 0;

end

for j = 1:4

if min(id\_or\_sharp{i}(1:19) == rfid\_table(1).(fields{j})) % If the RFID tag ID corresponds to an element in the lookup table

if t\_plot(j) % If the animation completed and was timed

t\_restart(j) = toc(t\_start(j)); % How long it took after the beginning of the animation for the robot to be read again

if green\_complete(j)

red\_increment(j) = round(red\_increment(j)\*(t\_plot(j)/t\_restart(j))); % Change increments according to how much sooner the...

green\_increment(j) = round(green\_increment(j)\*(t\_plot(j)/t\_restart(j))); % ... animation finished than the robot actually took

green\_frame(j) = 0;

green\_complete(j) = 0;

red\_frame(j) = 0;

red\_complete(j) = 0;

end

t\_plot(j) = 0; % Restart animation timer

else % If the robot was detected before...

if any(rfid(:,j)) && ishandle(p(j)) % ...its animation reached completion

if green\_frame(j) >= 6000 && any(rfid(:,j)) % A slight buffer so that getting multiple readings at a pass will not unduly speed up animation

green\_increment(j) = round(green\_increment(j)\*(22000/green\_frame(j))); % Adjust incrments according to how much longer the animation..

red\_increment(j) = round(red\_increment(j)\*(22000/green\_frame(j))); % ... had left when the robot was detected again

green\_frame(j) = 0;

green\_complete(j) = 0;

red\_frame(j) = 0;

red\_complete(j) = 0;

end

end

end

red\_complete(j) = 0; % Reset the completion variable for the animations

rfid(:,j) = 0; % Clear robot's past position

rfid(i,j) = 1; % Update robots position

fprintf('%s', robots{j})

fprintf('%s', ' detected at ')

fprintf('%s\n', readers{i})

elseif min(id\_or\_sharp{i}(1:19) == rfid\_table(2).(fields{j})) % If the second card of the two should be detected instead

if t\_plot(j)

t\_restart(j) = toc(t\_start(j));

if green\_complete(j)

red\_increment(j) = round(red\_increment(j)\*(t\_plot(j)/t\_restart(j)));

green\_increment(j) = round(green\_increment(j)\*(t\_plot(j)/t\_restart(j)));

green\_frame(j) = 0;

green\_complete(j) = 0;

red\_frame(j) = 0;

red\_complete(j) = 0;

end

t\_plot(j) = 0;

else

if any(rfid(:,j)) && ishandle(p(j))

if green\_frame(j) >= 6000 && any(rfid(:,j))

green\_increment(j) = round(green\_increment(j)\*(22000/green\_frame(j)));

red\_increment(j) = round(red\_increment(j)\*(22000/green\_frame(j)));

green\_frame(j) = 0;

green\_complete(j) = 0;

red\_frame(j) = 0;

red\_complete(j) = 0;

end

end

end

red\_complete(j) = 0;

rfid(:,j) = 0;

rfid(i,j) = 1;

fprintf('%s', robots{j})

fprintf('%s', ' detected at ')

fprintf('%s\n', readers{i})

end

end

end

end

end

for k = 1:4

for m = 1:4

if isred(mod(k-4,4)+1) && rfid(k,m) && ~red\_frame(m) % If animation is about to start

t\_start(m) = tic; % Start the timer

elseif ~isred(mod(k-4,4)+1) && rfid(k,m) && ~green\_frame(m)

t\_start(m) = tic;

end

end

end

id\_or\_sharp(i) = {1}; % Loop agian without a robot being detected

[red\_frame, p] = red\_1(rfid,red\_frame,p,red\_increment,isred,red\_complete);

[green\_frame, p] = green\_1(rfid,green\_frame,p,green\_increment,green\_complete,red\_complete);

[red\_frame, p] = red\_2(rfid,red\_frame,p,red\_increment,isred,red\_complete);

[green\_frame, p] = green\_2(rfid,green\_frame,p,green\_increment,green\_complete,red\_complete); % Step appropriate animation by 'increment'

[red\_frame, p] = red\_3(rfid,red\_frame,p,red\_increment,isred,red\_complete);

[green\_frame, p] = green\_3(rfid,green\_frame,p,green\_increment,green\_complete,red\_complete);

[red\_frame, p] = red\_4(rfid,red\_frame,p,red\_increment,isred,red\_complete);

[green\_frame, p] = green\_4(rfid,green\_frame,p,green\_increment,green\_complete,red\_complete);

for k = 1:4

for m = 1:4;

if (red\_frame(m) >= 4000 && rfid(k,m)) && ~isred(mod(k-4,4)+1) % If a red light animation has reached completion or the next light turns green

green\_frame(m) = red\_frame(m); % Since the 'green' animation should start where the 'red' one left off

id\_or\_sharp{k} = 0; % No need to loop through for that one anymore

red\_frame(m) = 0; % Restart that animation from the beginning for next time

red\_complete(m) = 1;

end

if ~isred(mod(k-4,4)+1) && ~red\_frame(m)

red\_complete(m) = 1;

elseif (isred(mod(k-4,4)+1) || red\_frame(m))

red\_complete(m) = 0;

end

if green\_frame(m) >= 22000 && rfid(k,m)

t\_plot(m) = toc(t\_start(m)); % Determine how long the animation took

id\_or\_sharp{k} = 0;

green\_frame(m) = 0;

rfid(k,m) = 0;

green\_complete(m) = 1;

end

end

end

drawnow

pause(1/10)

end

Sample Red Light Animation Program Listing

function [frame, p] = red\_1(rfid, frame, p, increment, isred, iscomplete)

for k = 1:4

for m = 1:4

if rfid(1,m) && iscomplete(m)

return

elseif ~any(rfid(1,:)) || (frame(m) >= 4000) % If the animation should not be run

return

end

end

end

x0 = [7300,6300,6300,7300]; % Starting position for the robot patch

y0 = [1350,1350,600,600];

color(1) = 'r';

color(2) = 'g'; % Color code to differentiate robots

color(3) = 'm';

color(4) = 'c';

for j = 1:4;

if rfid(1,j) % Determine which robot was detected

if frame(j) == 0 && iscomplete(j);

if ishandle(p(j))

delete(p(j))

end

p(j) = patch(x0, y0, color(j)); % Color the patch accordingly

end

end

end

for j = 1:4;

if rfid(1,j)

if frame(j) <= 2000 && frame(j) >= 0

set(p(j), 'visible', 'off');

p(j) = patch(x0+frame(j), y0, color(j));

frame(j) = frame(j) + increment(j);

end

if frame(j) > 2000 && frame(j) <= 4000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y, color(j));

rotate(p(j), [0,0,1], 0.045\*increment(j), [(max(x)+min(x))/2,(max(y)+min(y))/2,0]);

frame(j) = frame(j) + increment(j);

end

% if frame(j) <= 10000 && frame(j) > 8000

% set(p(j), 'visible', 'off');

% a = get(p(j));

% x = a.Vertices(:,1)';

% y = a.Vertices(:,2)';

% p(j) = patch(x-increment(j), y, color(j));

% frame(j) = frame(j) + increment(j);

% end

end

end

end

Sample Corresponding Green Light Animation Program Listing

function [frame, p] = green\_1(rfid, frame, p, increment, green\_complete, red\_complete)

for k = 1:4

for m = 1:4

if rfid(1,m) && (green\_complete(m) || ~red\_complete(m))

return

elseif ~any(rfid(1,:))% || (~red\_complete(m))

return

end

end

end

x0 = [7300,6300,6300,7300];

y0 = [1350,1350,600,600];

color(1) = 'r';

color(2) = 'g';

color(3) = 'm';

color(4) = 'c';

for j = 1:4;

if rfid(1,j)

if frame(j) == 0;

if ishandle(p(j))

delete(p(j))

end

p(j) = patch(x0, y0, color(j));

end

end

end

for j = 1:4;

if rfid(1,j)

set(p(j), 'visible', 'off');

if frame(j) <= 2000 && frame(j) >= 0

p(j) = patch(x0+frame(j), y0, color(j));

frame(j) = frame(j) + increment(j);

end

if frame(j) > 2000 && frame(j) <= 4000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y, color(j));

rotate(p(j), [0,0,1], 0.045\*increment(j), [(max(x)+min(x))/2,(max(y)+min(y))/2,0]);

frame(j) = frame(j) + increment(j);

end

if frame(j) <= 8000 && frame(j) > 4000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y+increment(j), color(j));

frame(j) = frame(j) + increment(j);

end

if frame(j) > 8000 && frame(j) <= 10000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y, color(j));

rotate(p(j), [0,0,1], 0.045\*increment(j), [(max(x)+min(x))/2,(max(y)+min(y))/2,0]);

frame(j) = frame(j) + increment(j);

end

if frame(j) > 10000 && frame(j) <= 18000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x-increment(j), y, color(j));

frame(j) = frame(j) + increment(j);

end

if frame(j) > 18000 && frame(j) <= 20000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y, color(j));

rotate(p(j), [0,0,1], 0.045\*increment(j), [(max(x)+min(x))/2,(max(y)+min(y))/2,0]);

frame(j) = frame(j) + increment(j);

end

if frame(j) <= 22000 && frame(j) > 20000

set(p(j), 'visible', 'off');

a = get(p(j));

x = a.Vertices(:,1)';

y = a.Vertices(:,2)';

p(j) = patch(x, y-increment(j), color(j));

frame(j) = frame(j) + increment(j);

end

end

end

end

Simplified MATLAB Program for Controlling Lights

clear all; clc; clf; close all;

%% Initialize plot/animation

figure(1)

hold all

xlim([0, 10000]);

ylim([0, 10000]);

axis equal

rectangle('position', [2000, 2000, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % Plot

rectangle('position', [2100, 2100, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [2000, 5500, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % all

rectangle('position', [2100, 5600, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [5500, 5500, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % the

rectangle('position', [5600, 5600, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [5500, 2000, 2500, 2500], 'facecolor', [(248/255), (222/255), (126/255)]); % boxes

rectangle('position', [5600, 2100, 2300, 2300], 'facecolor', [1, 1, 1]);

rectangle('position', [6000, 0, 500, 2000], 'facecolor','k')

r(1) = rectangle('position', [5850, 1000, 150, 1000], 'facecolor', 'r');

g(1) = rectangle('position', [5850, 0, 150, 1000], 'facecolor', 'g');

rectangle('position', [8000, 6000, 2000, 500], 'facecolor', 'k')

r(2) = rectangle('position', [8000, 5850, 1000, 150], 'facecolor', 'r');

g(2) = rectangle('position', [9000, 5850, 1000, 150], 'facecolor', 'g');

rectangle('position', [3500, 8000, 500, 2000], 'facecolor', 'k')

r(3) = rectangle('position', [4000, 8000, 150, 1000], 'facecolor', 'r');

g(3) = rectangle('position', [4000, 9000, 150, 1000], 'facecolor', 'g'); % Plot ladders and their lights

rectangle('position', [0, 3500, 2000, 500], 'facecolor', 'k')

r(4) = rectangle('position', [1000, 4000, 1000, 150], 'facecolor', 'r');

g(4) = rectangle('position', [0, 4000, 1000, 150], 'facecolor', 'g');

for z = 1:4 % All lights are initially off

set(g(z), 'visible', 'off')

set(r(z), 'visible', 'off')

end

rectangle('position', [6500, 0, 750, 2000], 'facecolor', 'y');

text(6760, 1000, '1', 'fontsize', 14)

rectangle('position', [8000, 6500, 2000, 750], 'facecolor', 'y');

text(9000, 6900, '2', 'fontsize', 14)

rectangle('position', [2750, 8000, 750, 2000], 'facecolor', 'y') % Plot and enumerate RFID speed bumps

text(3010, 9000, '3', 'fontsize', 14)

rectangle('position', [0, 2750, 2000, 750], 'facecolor', 'y')

text(1000, 3100, '4', 'fontsize', 14)

rectangle('position', [3000, 4500, 4000, 1000],...

'facecolor', [(215/255),(215/255),(215/255)]) % Plot bridge

p(1:4) = patch([0,0,0,0],[0,0,0,0],'k'); % Preallocate patches that will represent robots

red\_frame = zeros(1,4); % Initialize frame of red light animations

red\_increment(1:4) = 100\*ones(1,4); % How far the animation will proceed each step

green\_frame = zeros(1,4);

green\_increment(1:4) = 100\*ones(1,4);

t\_plot = zeros(1,4); % Initialize variables to be used to time animaions for speed correction

t\_restart = zeros(1,4);

red\_complete = zeros(1,4); % Logic values representing whether an animation is finished or not

green\_complete = zeros(1,4);

green\_start = zeros(1,4);

%% Initialize RFID lookup table and RFID readers

delete(instrfind); % Close any serial device communications left open

id\_or\_sharp = {0,0,0,0}; % Create variable in which to store data from Arduino circuit

rfid = zeros(4);

fields = {'robot1', 'robot2', 'robot3', 'robot4'};

robots = {'Robot #1', 'Robot #2', 'Robot #3', 'Robot #4'};

readers = {'Reader #1', 'Reader #2', 'Reader #3', 'Reader #4'};

reader(1) = serial('COM8','BaudRate',9600);

reader(2) = serial('COM7','BaudRate',9600); % Connect to the COM ports

reader(3) = serial('COM6','BaudRate',9600);

reader(4) = serial('COM9','BaudRate',9600);

for i = 1:4;

fopen(reader(i)); % Open communication on the COM ports

flushinput(reader(i)); % Clear Arduinos' serial buffers

end

rfid\_table = struct('robot1', {'0xFD 0x8C 0x82 0x9A','0x95 0x6C 0x59 0x6F'},...

'robot2', {'0x7D 0xFE 0x88 0x9A','0x5D 0xEE 0x82 0x9A'},...

'robot3', {'0x25 0x68 0xA2 0x51','0x35 0xE9 0x7D 0x33'},... % Robot lookup table

'robot4', {'0xED 0x1E 0x86 0x9A','0x2D 0x5A 0x83 0x9A'});

lights = zeros(4,1); % The state of each light set is represented by one element of the variable

isred = zeros(1,4);

q = 1;

while q < 5; % For some reason, this loop allows lights to be turned on...

fscanf(reader(q)); % ...(initial conditions) without an RFID tag yet being scanned.

lights(2) = 2;

set(g(2),'visible','on')

set(r(2),'visible','off')

fwrite(reader(2),lights(2))

lights(4) = 1;

set(g(4),'visible','off')

set(r(4),'visible','on')

fwrite(reader(4),lights(4))

isred(4) = 1;

q = q+1;

end

%% Control and plot lights

while 1

for i = 1:4

if reader(i).BytesAvailable % If there is anything in circuits' buffers

id\_or\_sharp(i) = {fscanf(reader(i))}; % Store buffer data (SHARP or PN532) in a variable

flushinput(reader(i)) % Clear buffer to prevent unnecessary entry into loops

if length(id\_or\_sharp{i}) == 21 || (length(id\_or\_sharp{i}) == 1 && id\_or\_sharp{i} == 1); % If data is an RFID tag ID or the loop should be repeated

if i == 2 % Should the robot cross the bridge?

lights(2) = 2;

set(g(2),'visible','on');

set(r(2),'visible','off');

fwrite(reader(2),lights(2));

isred(2) = 0;

lights(4) = 1;

set(g(4),'visible','off');

set(r(4),'visible','on');

fwrite(reader(4),lights(4));

isred(4) = 1;

elseif i == 4

lights(2) = 1;

set(g(2),'visible','off');

set(r(2),'visible','on');

fwrite(reader(2),lights(2));

isred(2) = 1;

lights(4) = 2;

set(g(4),'visible','on');

set(r(4),'visible','off');

fwrite(reader(4),lights(4));

isred(4) = 0;

end

end

end

end

drawnow

end