Collision Avoidance Robot

By: Salvatore Carollo; Jessica Catelotti; Antu Das; Rizvo Memisevic; Konrad Mozdzen; Jerry Stuart; Peter Thomsen;

For: ETC 423-01A

Date: 12/14/2015

Overview

Abstract

- **Primary Task:**
  - Design an object avoiding robot with the use of IR Sensors and the Arduino Board
- **Secondary Tasks:**
  - Add an alternate path
  - Incorporate the use of push buttons for path selection
  - Use LED’s to communicate to the user which path has been selected

Real World Implementation

- Robotic Tour Guide

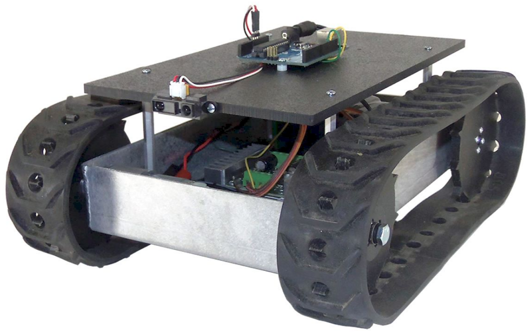
The same concept of collision avoidance can be used to design a robot that leads a group of people through a museum. This technology is currently being used in the world today for that very purpose.

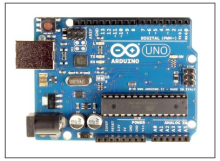
Robot Specifications

- **Dimensions:** ~10.75" Wide x ~12.5" long and ~4.75" tall
- **Capacity:** Handles up to 15lbs of additional payload in most terrains. (Different option motors selected, standard option gear motors only handle 5lbs additional payload.)
- **Speed:** Variable speed and direction differential drive/steering. The motors drive this robot up to 55 feet per minute (different motors can be selected as options).
- **Weight:** The total weight of this configuration is ~6 lbs.

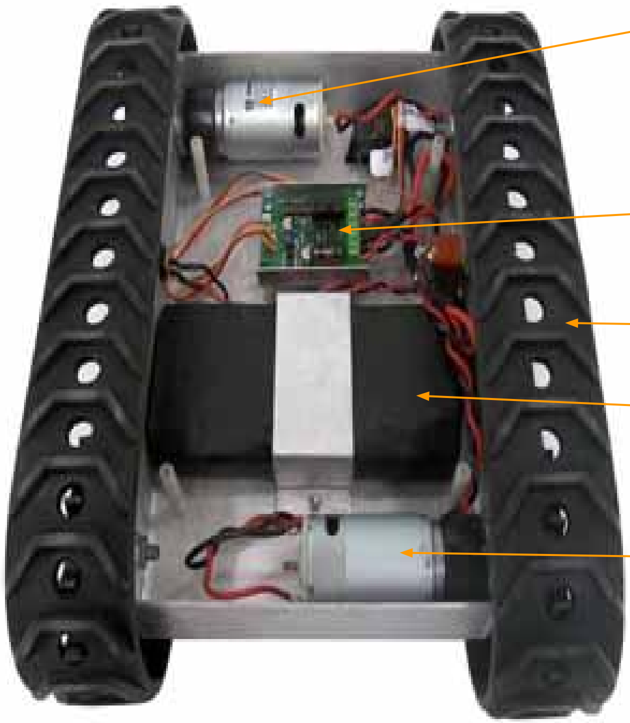
Components Used

- Programmable MLT-JR Tracked Development Robot
- Power Supply
- Arduino Uno Board
- Infrared Sensor
- Push Buttons
- LEDs

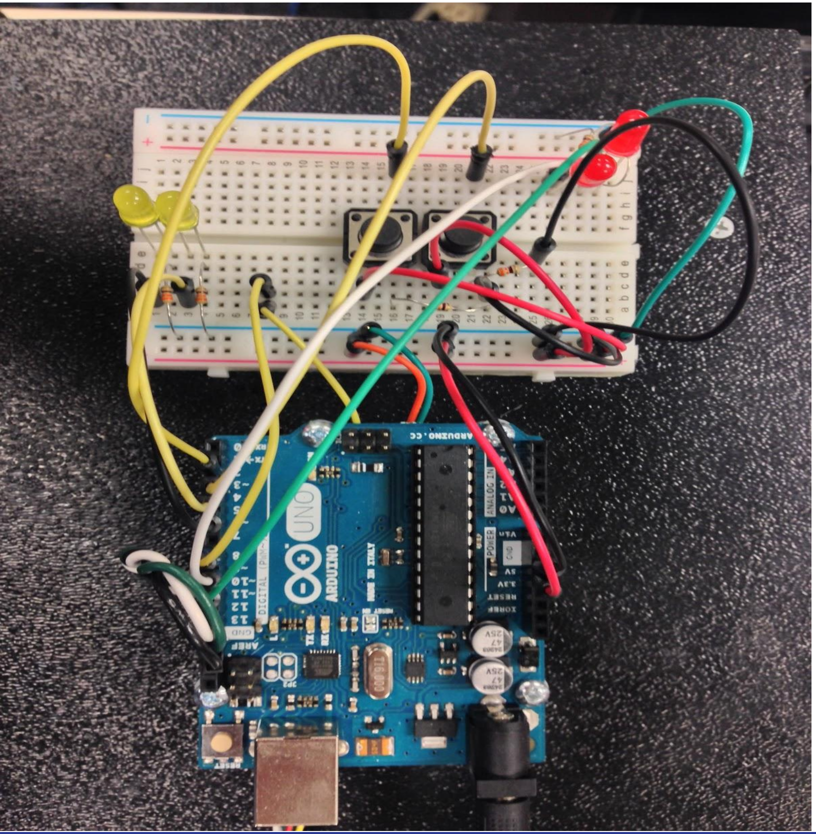




Components Inside of the Robot


Front Motor
Sabertooth Dual 12A RC Motor Driver
Molded Spliceless Tracks and Wheel Set
Battery Pack
Rear Motor

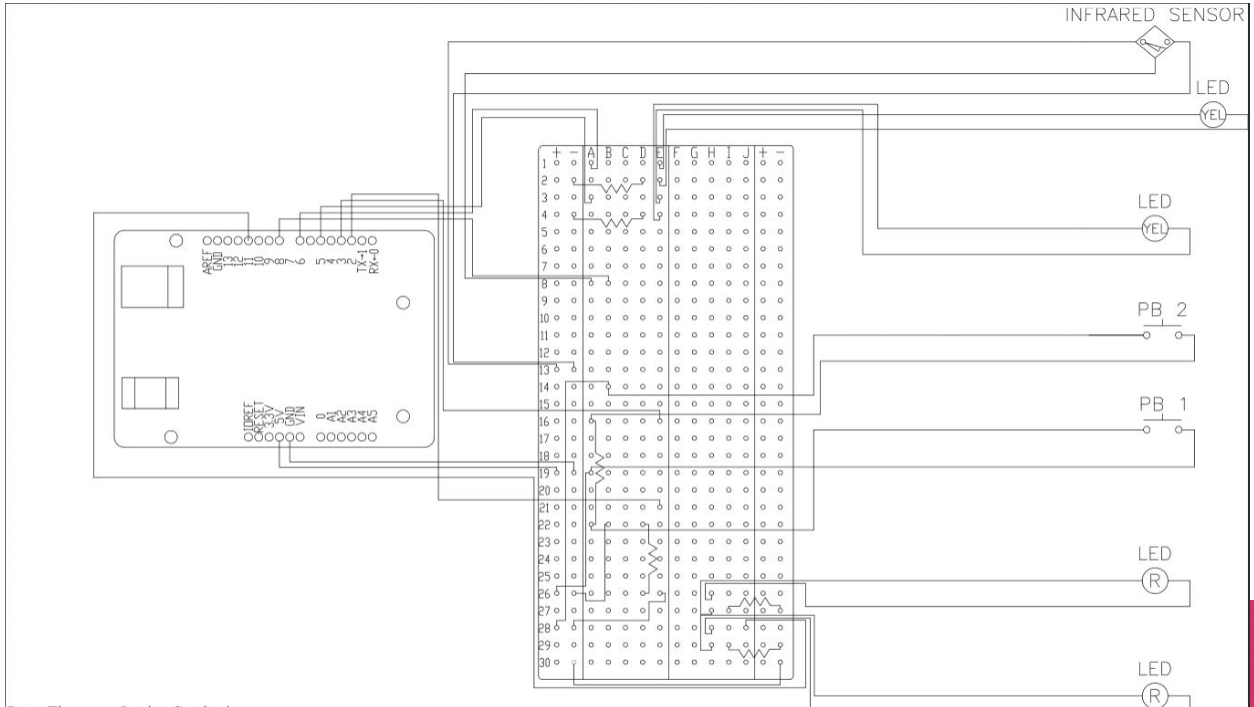
Arduino Board, Wiring, and Connections



Pin	Connected to
2	Switch 1
3	Switch 2
4	LED 1
5	LED 2
8	Sensor

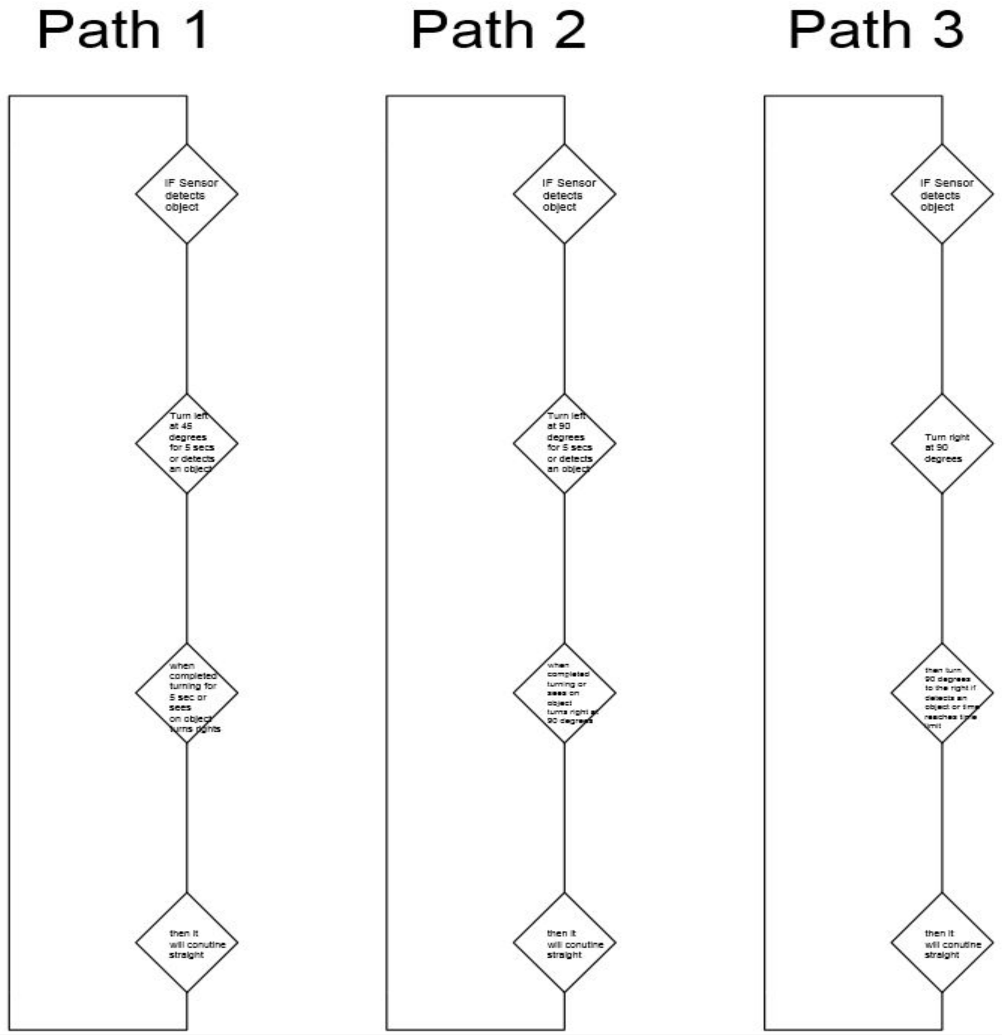
Component	Connected to
LED 1	Resistor 1
LED 2	Resistor 2
Resistor 1&2	Ground
Resistor 3&4	Push Buttons
Push Buttons	Power
Resistor 3&4	Ground
Sensor	Ground
Sensor	Power

Wiring Diagram



Peter Thomsen, Jessica Catelotti,
Salvatore Carollo, Antu Das,
Rizvo Memisevic, Konrad Mozden,
Jerry Stuart

Flow Chart


Path 1
Path 2
Path 3
IF Sensor detects object
Turn left at 45 degrees for 5 secs or detects an object
when completed turning for 5 sec or sees on object turns rights
then it will conutine straight
IF Sensor detects object
Turn left at 90 degrees for 5 secs or detects an object
when completed turning or sees on object turns right at 90 degrees
then it will conutine straight
IF Sensor detects object
Turn right at 90 degrees
then turn 90 degrees to the right if detects an object or time reaches time limit
then it will conutine straight

Salvatore Carollo, Jessica Catelotti,
Antu Das, Rizvo Memisevic,
Konrad Mozdzen,Jerry Stuart,
Peter Thomsen

Understanding the Program

```
// Libraries
#include <SoftwareSerial.h>
#include <Sabertooth.h>
// Motor Controller
SoftwareSerial SWSerial(NOT_A_PIN, 13); // RX on no pin (unused), TX on
pin 13 (to S1).
Sabertooth ST(128, SWSerial);                  // Address 128, and use SWSerial
as the serial port.
// Global Constants
const int buttonPath = 2;   // the pin that the pushbutton for path selection
is attached to
const int buttonConf = 3;   // The pin that the pushbutton for path
conformation is attached to
const int ledZero = 4;                 // The pin that LED for 2^0 is attached to
const int ledOne = 5;                  // The pin that LED for 2^1 is attached to
const int IR = 8;         // IR sensor pin
const int flashingLED_1 = 9;
const int flashingLED_2 = 10;
// Global Variables
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;                   // current state of the button
int lastButtonState = 0;               // previous state of the button
int IR_cnt = 0;           // counter used in object detection
int endReached = 0;                    // trigger used to determine if in a corner
```

```
void setup()
{
  SWSerial.begin(9600);   // Serial communicaiton to the motor
controller
  delay(200);            // Give the motor controller time to boot
  ST.autobaud();         // The motor controller configures it's baud
rate to
                         // match the first character sent to it. The
autobaud
                         // function sends 0xAA (0b 1010 1010).
  // initializing I/O:
  pinMode(buttonPath, INPUT);
  pinMode(buttonConf, INPUT);
  pinMode(ledZero, OUTPUT);
  pinMode(ledOne, OUTPUT);
  pinMode(flashingLED_1, OUTPUT);
  pinMode(flashingLED_2, OUTPUT);
}
void loop()
{
  pathSelect();
  pathRun(buttonPushCounter);
}
void pathSelect()
```

Understanding the Program

```
* Function runs until the path conformation button (buttonConf) is pressed.  Each time the
 * path selection button (buttonPath) is pressed, a path selection counter (buttonPushCounter)
 * is incremented.  There are only three valid path entries therefore if the path selection
 * counter (buttonPushCounter) is out of range, counter is reset to 1.
{
  while(!digitalRead(buttonConf))
  {
        buttonState = digitalRead(buttonPath);    // Update pushbutton input
        if(buttonState != lastButtonState)        // If pushbutton input has changed from previously stored state
        {
        if(buttonState == HIGH)
        // If pushbutton is currently pressed (Leading edge trigger)
        {
        buttonPushCounter++;                      // Increment path selection counter
        }
        delay(50);                       // Delay for debouncing
        }
        lastButtonState = buttonState;            // Update/store pushbutton state
        if(buttonPushCounter == 4)                // True if path selection is out-of-range
        {
        buttonPushCounter = 1;                    // Reset path selection counter
        }
        updateLED(buttonPushCounter);             // Update LEDs with path selection counter value
  }
}
```

Understanding the Program

```
void updateLED(int value)
/*
 * This function displays the given value in binary on LEDs ledZero and ledOne.  Expected
 * passed value range is between 0 and 3.
 */
{
   if(value == 0)                  // Display a binary 0 on LEDs
   {
     digitalWrite(ledZero, LOW);
     digitalWrite(ledOne, LOW);
   }
   else if(value == 1)             // Display a binary 1 on LEDs
   {
     digitalWrite(ledZero, HIGH);
     digitalWrite(ledOne, LOW);
   }
   else if(value == 2)             // Display a binary 2 on LEDs
   {
     digitalWrite(ledZero, LOW);
     digitalWrite(ledOne, HIGH);
   }
   else                            // value == 3, display a binary 3 on LEDs
   {
     digitalWrite(ledZero, HIGH);
     digitalWrite(ledOne, HIGH);
   }
}
```

Understanding the Program

```
void pathRun(int value)
/*
 * Calls path function that corresponds to the value given when called.  After completion,
 * function pathSuccess is called signaling the completion of path function called.
 */
{
  if(value == 1)
  {
    pathOne();
    pathSuccess();
  }
  else if(value == 2)
  {
    pathTwo();
    pathSuccess();
  }
  else
  {
    pathThree();
    pathSuccess();
  }
}
```

Understanding the Program

```
void pathSuccess()
/*
 * This function flashes the LEDs as a visual conformation that a path has ran to completion.
 * The number 3 (in binary) is flashed three times, pushbutton counter reset.
 */
{
  updateLED(0);
  delay(50);
  updateLED(3);
  delay(250);
  updateLED(0);
  delay(50);
  updateLED(3);
  delay(250);
  updateLED(0);
  delay(50);
  updateLED(3);
  delay(250);
  updateLED(0);
  buttonPushCounter = 0;
}
```

Understanding the Program

```
void tourGuideDesc()
/*
 * This function will be used to indicate that our robot has stopped to provide a tour guide's
 * description of the area.  If we have time, the LEDs could be replaced by an audio-playback
 * device.
 *
 * CHANGES - renamed function as its been re-purpose, replaced while-statement with an incrementing
 *           counter to repeat this function for a set amount of time.
 */
{
  int descLength = 0;
  while(descLength < 5000)     // Change this value to increase/decrease function runtime
  {
    digitalWrite(9, HIGH);     // turn the LED on (HIGH is the voltage level)
    digitalWrite(10, HIGH);
    delay(500);                // wait for a second
    digitalWrite(9, LOW);      // turn the LED off by making the voltage LOW
    digitalWrite(10, LOW);
    delay(500);
    descLength++;
  }
/*  else
  {
    digitalWrite(9,LOW);
    digitalWrite(10,LOW);
  } */
}
```

Understanding the Program

```
void pathOne()
/*
 * This function is kinda broken because it runs indefinitely.  To "fix" this a counter must
 * be added which increments after each wall/bend has been avoided.  Once all walls/bends
 * have been passed, keep moving until classroom doorway is reached (C014) using a delay.
 */
{
  IR_cnt = 0;   // Counter used to determine whether object is a wall or not

  while(1)
  {
    if(digitalRead(IR))   // If IR = 1 (TRUE), no object detected
    {
      ST.drive(128);    // Drive forward at full speed
      ST.turn(0);
    }
    else                // Else, IR = 0, object detected
    {
      ST.drive(0);      // Stop both motors
      IR_cnt++;
      delay(250);       // wait 250ms
    }

    if(IR_cnt == 20)    // Object still remains after 5sec (20*250ms=5sec)
    // This IF-statement will be used to travel around the walls/bends in the hallway
    {
      // Here we are trying to turn 45 degrees to the left at full speed
 ST.turn(-128);
      delay(500);       // Modifier for distance travelled
      ST.drive(0);

      // Now we are ready to drive along the front of the object until adjacent wall
detected
      ST.drive(128);
      ST.turn(0);

      while(digitalRead(IR))
      {
        // Once adjacent wall detected (IR = 0), while loop is exited.
      }
      ST.drive(0);

      // By this point the object should be cleared, we need to orient ourselves
before driving forward
      // The following code should mirror exactly our first turn at the beginning of
IF-statement
      ST.turn(128);
      delay(500);
      ST.drive(0);

      // We are now clear of the wall/bend and can now continue.
      IR_cnt = 0;       // Reset conditional counter
    } // if(IR_cnt == 20) END
  }  // while(1) END
}    // pathOne END
```

Understanding the Program

```
void pathTwo()
/*
 * This function is kinda broken because it runs indefinitely.  To "fix" this a counter must
 * be added which increments after each wall/bend has been avoided.  This path is the same as
 * path one but runs in reverse
 */
{
  IR_cnt = 0;   // Counter used to determine whether object is a wall or not

  while(1)
  {
    if(digitalRead(IR))   // If IR = 1 (TRUE), no object detected
    {
      ST.drive(128);    // Drive forward at full speed
      ST.turn(0);
    }
    else                // Else, IR = 0, object detected
    {
      ST.drive(0);        // Stop both motors
      IR_cnt++;
      delay(250);         // wait 250ms
    }

    if(IR_cnt == 20)    // Object still remains after 5sec (20*250ms=5sec)
    // This IF-statement will be used to travel around the walls/bends in the hallway
    {
      // Here we are trying to turn 90 degrees to the left at full speed
      ST.turn(128);
 delay(1000);           // Modifier for distance travelled
      ST.drive(0);

      // Now we are ready to drive along the front of the object until adjacent wall
detected
      ST.drive(128);
      ST.turn(0);

      while(digitalRead(IR))
      {
        // Once adjacent wall detected (IR = 0), while loop is exited.
      }
      ST.drive(0);

      // By this point the object should be cleared, we need to orient ourselves before
driving forward
      // The following code should mirror exactly our first turn at the beginning of
IF-statement
      ST.turn(-128);
      delay(1000);
      ST.drive(0);

      // We are now clear of the wall/bend and can now continue.
      IR_cnt = 0;         // Reset conditional counter
    } // if(IR_cnt == 20) END
  }   // while(1) END
}     // pathTwo END
```

Understanding the Program

```
void pathThree()
/*
 * This path is our "snake n' clear", the robot snakes back-and-forth along the room in a
 * rectangular zagging pattern.  Once the robot has driven into a corner, we have determined
 * the room has been clear and function ends.
 */
{
  // Initializing function variables used in this path
  IR_cnt = 0;                              // Counter used to determine whether object is a wall or not
  int firstTurn = 0, corner = 0;      // Used for right turn completion and corner detection
  digitalWrite(flashingLED_1, LOW);

  delay(1000);                    // Short delay before movement begins

  while(!corner)
  // If we determined that a corner has been reached, exit while-loop
  {
    while(firstTurn == 0 && corner == 0)
    {
      if(firstTurn == 0 && corner == 0)
      {

        if(digitalRead(IR))           // If IR = 1 (TRUE), no object detected
        /*
        * Robot will continue foward as long as path is clear, once an object is detected,
        * the robot will stop, INC a counter and continue forward once object is cleared OR
        * next IF statement becomes true.
        */
```

Understanding the Program

```
    {
        ST.drive(128);
        ST.turn(0);
    }
    else              // Else, IR = 0, object detected
    {
        ST.drive(0);          // Stop both motors
        IR_cnt++;
        delay(250);           // wait 250ms
    }

    if(IR_cnt == 10)          // Object still remains after 2.5sec (10*250ms=2.5sec)
    /*
    * Its been determined that a wall has been reached.  Robot will stop foward progress
    * before making two 90-degree turns before continuing to clear the room in the
opposite
    *  direction.
    */
    {
        // turning 90 degrees for 500ms
        ST.drive(0);
        ST.turn(128);
        delay(900);

        // drive forward a little bit
        ST.turn(0);
```

```
if(!digitalRead(IR))
        {
        corner = 1;
        }
        else
        {
        ST.drive(128);
        ST.turn(0);
      delay(500);

        // turning 90 again = 180 degree turn
```

Understanding the Program

```
            ST.drive(0);
            ST.turn(128);
            delay(900);
            ST.turn(0);
            IR_cnt = 0;             // counter reset for next wall detection process
          firstTurn++;
            digitalWrite(flashingLED_1, HIGH);
            delay(1000);
            } // else END
    }  // if(IR_cnt == 10) END
   }       // if(firstTurn == 0 && corner == 0) END
  }         // while(firstTurn == 0 && corner == 0) END
  /*
   * The previous lines of code (above) will repeat but this time once the opposite  wall
   * is reached, the robot will make two 90-degree turns to the left.
   */

  while(firstTurn == 1 && corner == 0)
  {
   if(firstTurn == 1 && corner == 0)
   {
    if(digitalRead(IR))
    {
            ST.drive(128);
            ST.turn(0);
    }
```

```
    else
        {
            ST.drive(0);
            IR_cnt++;
            delay(250);
        }

        delay(500);

        if(IR_cnt == 10)
        {
            // turning 90 degrees for 500ms
            ST.drive(0);
            ST.turn(-128);
            delay(900);

            // drive forward a little bit
```

Understanding the Program

```
            ST.turn(0);
            if(!digitalRead(IR))
            {
            corner = 1;
            }
            else
            {
            ST.drive(128);
            ST.turn(0);
            delay(500);

            // turning 90 again = 180 degree turn
            ST.drive(0);
            ST.turn(-128);
            delay(900);
            ST.turn(0);
            IR_cnt = 0;          // counter reset for next wall detection process
            firstTurn--;
            digitalWrite(flashingLED_1, LOW);
            delay(1000);
         } // else END
        }  // if(IR_cnt == 10) END
       }    // if(firstTurn == 1 && corner == 0) END
      }     // while(firstTurn == 1 && corner == 0) END
     }      // while(!corner) END
    }       // pathThree END
```

Milestone: Testing Functionality of the Robot

A. Upload the provided default test code to the Arduino Board

B. Ensure that the robot works properly

(Left Turn, Right Turn, Forward, and Reverse)

C. Begin writing our own code

Milestone: Testing Components

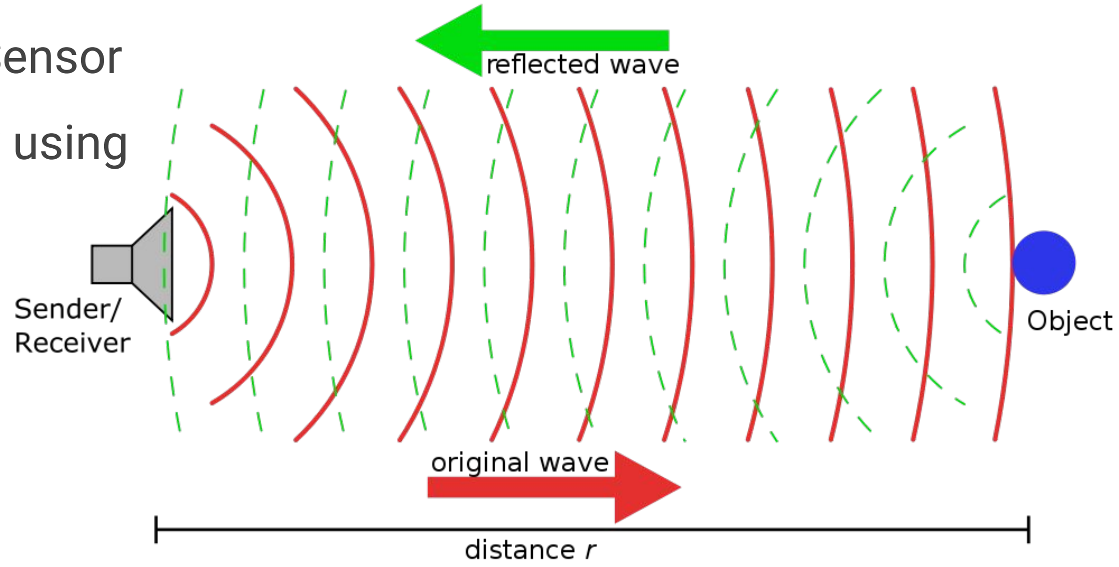
A. Write code for the Infrared Sensor

B. Track the sensor's status by using a Serial Monitor

0 = Object Detected

1 = No Object Detected

C. Ensure that the sensor works properly and attach it to the robot



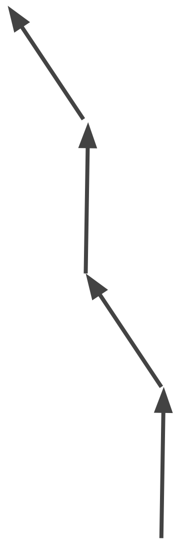
Milestone: Testing Arduino in Combination with Components

A. Testing whether the sensor is able to detect objects that are a certain distance away from it and return the data back to the arduino.
B. Testing if the two push buttons are functional and do what they were programmed to do.
C. Observing whether or not the LEDs and resistors are operational and perform their required tasks.
D. Testing the robot’s internal components performance, such as the battery pack and the motors.

Milestone: Demonstrating the Project as a Whole

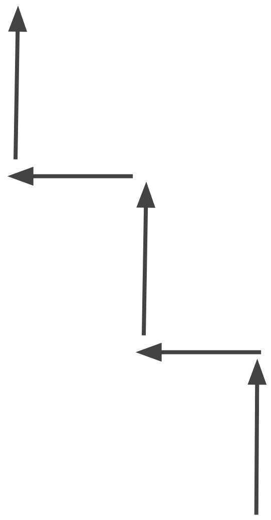
Path 1

⚫ 🟡



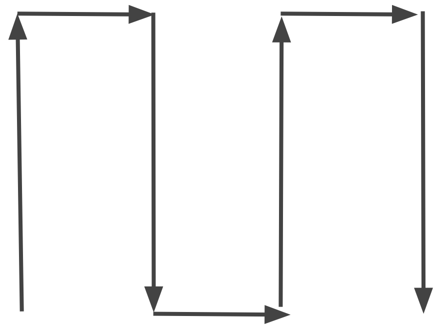
Path 2

🟡 ⚫



Path 3

🟡 🟡

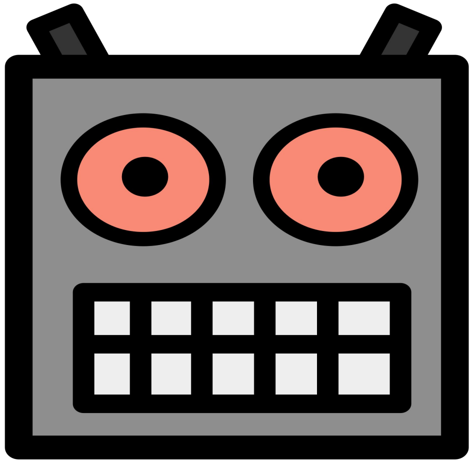


*LEDs function in Binary in respect to the selected paths

⚫	LED OFF
🟡	LED ON

Problems and Solutions: The “brief” summary

- Once upon a time - It all started with a tardy robot….
- Broken on Day 1!
- The demo code and the IDE library
- The 1st object detection malfunction
- The battle with the Encoder and our ultimate loss
- The 2nd object detection malfunction
- The alignment issue
- Paths and pushbuttons



Problems and Solutions: The “brief” summary

- The broken wheel: Part 1
- Death of the speaker idea
- Path selection and continued alignment issues
- The broken wheel: Part 2
- Continued path selection issues
- The disco bot!
- Our ultimate loss to the wheels