Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

Final Project

Introduction

There is increasing demand for both autonomous devices and convenient human-computer interaction. Large remote controls or computers for controlling devices is less desirable as people move to cellphones or even camera-based operation of devices. The goal of our project was to build an obstacle-avoiding, Bluetooth-controlled robot car. How this is achieved shall be described in the next section.

Description

To build our obstacle-avoiding, Bluetooth-operated car we used an Arduino UNO for the microcontroller. Obstacle-avoidance/detection was achieved using two ultrasonic distance sensors (a.k.a PING sensors). One distance sensor was forward facing while the other distance sensor was at a right angle to the first one and faced to the right. This gave us readings in two directions at any given moment and helped in making more robust navigation decisions. We used a Bluetooth Serial Pass-Through Module that would enable communication between a Bluetooth-enabled device and the Arduino UNO. This allowed us to connect a cellphone to the Bluetooth module and send commands to the microcontroller. The car would then move with the ultrasonic distance sensors alerting the microcontroller whenever an object was detected within a given distance.

The two motors of the car (one for each wheel) were controlled using a dual H-bridge chip that was also wired to the microcontroller. Based on either the user input or the detection of an obstacle, the microcontroller would send commands on how the wheels should move and direct the car. The entire system was powered by a 9V battery.

Operation

The operation of the car has been made simple. Once the car is turned on, the red LED on the Bluetooth module blinks until a device is connected. We used an existing Android app called “Bluetooth terminal” that simply creates a text interface. Using an Android cellphone, we connected to the Bluetooth module on our car and typed commands into the “Bluetooth terminal” app. Communication from the Arduino UNO to our phone is also displayed inside the app.

Our car has three modes that can be selected by sending a “0”, “1”, “2”. The car starts in idle mode which can also be returned to at any point by sending a 0 through the app. Sending a 1 puts the car in automatic mode, where the car drives around and avoids obstacles. Sending a 2 puts the car in manual mode. In manual mode, the car no longer avoids obstacles because the user can choose how to drive the car. A “w” moves the car forward, “a” moves the car left, “d” moves the car right, and “s” has the car reverse straight backwards.

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

Sensors and Actuators

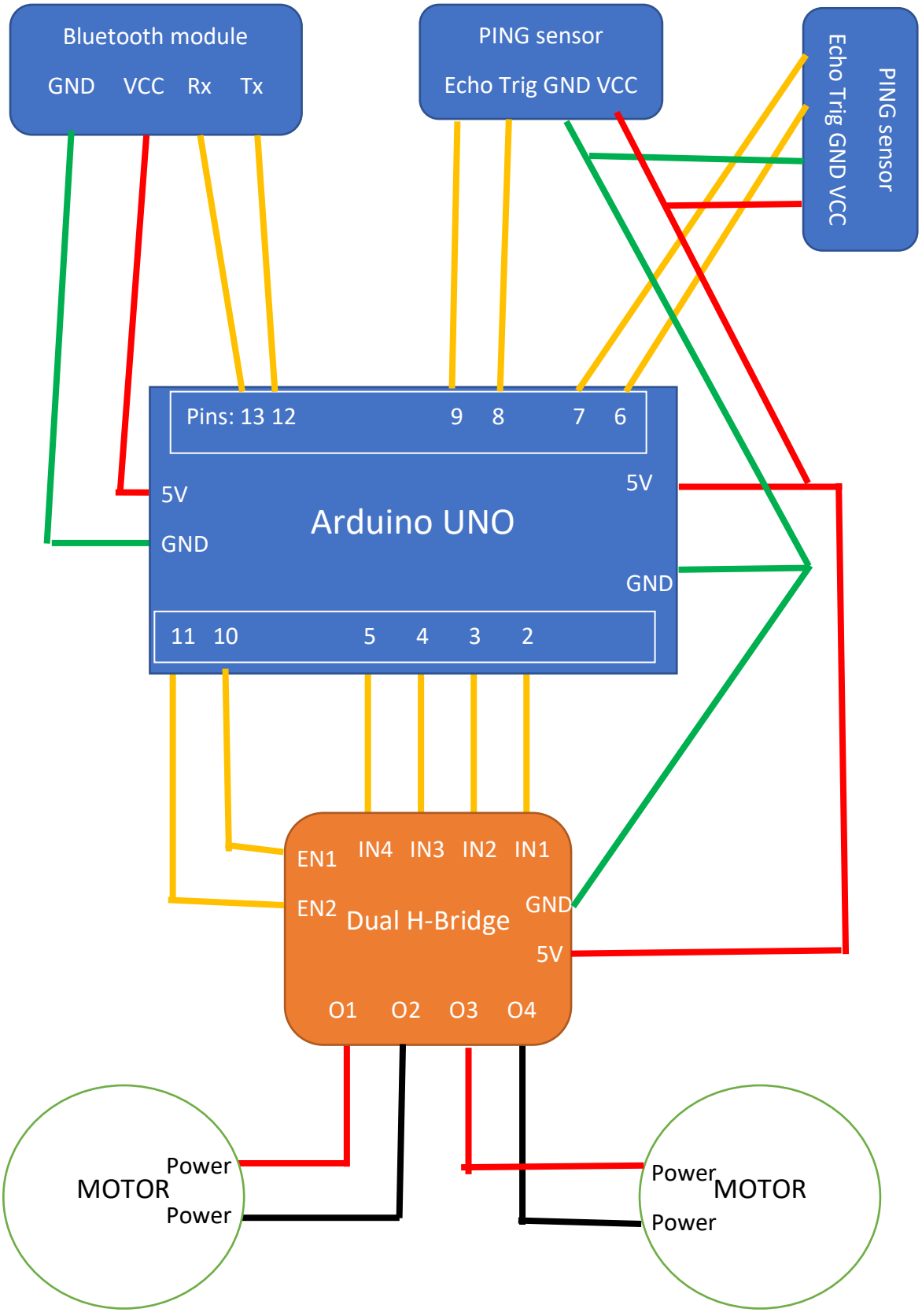
1. Ultrasonic Distance Sensors (x2) – emits an ultrasound that then moves through the air and can potentially bounce off objects. If any sound bounces back to the receiver of the distance sensor, the distance of the object can be calculated based on the speed of sound in air and the time it took to emit and received the sound.
   We specifically used knockoff HC-SR04 ultrasonic distance sensors.
2. Bluetooth Serial Pass-Through Module – a Bluetooth module that can send and receive data. Data is received from a Bluetooth enabled devices given to the Arduino UNO and the data can also be given to the module by the UNO and sent to the connected Bluetooth-enabled device.
   For this we used an HC-06 Bluetooth module.
3. DC Gear Motor 48:1 ratio (x2) – a simple DC gear motor that takes 3 to 9V.
   These motors are very cheap and were bought as part of set that included a chassis and wheels.

Circuit Diagrams (Pin Layout moved around for cleaner diagram)

**Note: The Arduino UNO only has one 5V that was used for all of the 5V in diagram. We have multiple 5V locations to clean the diagram, real wiring is more complicated.*

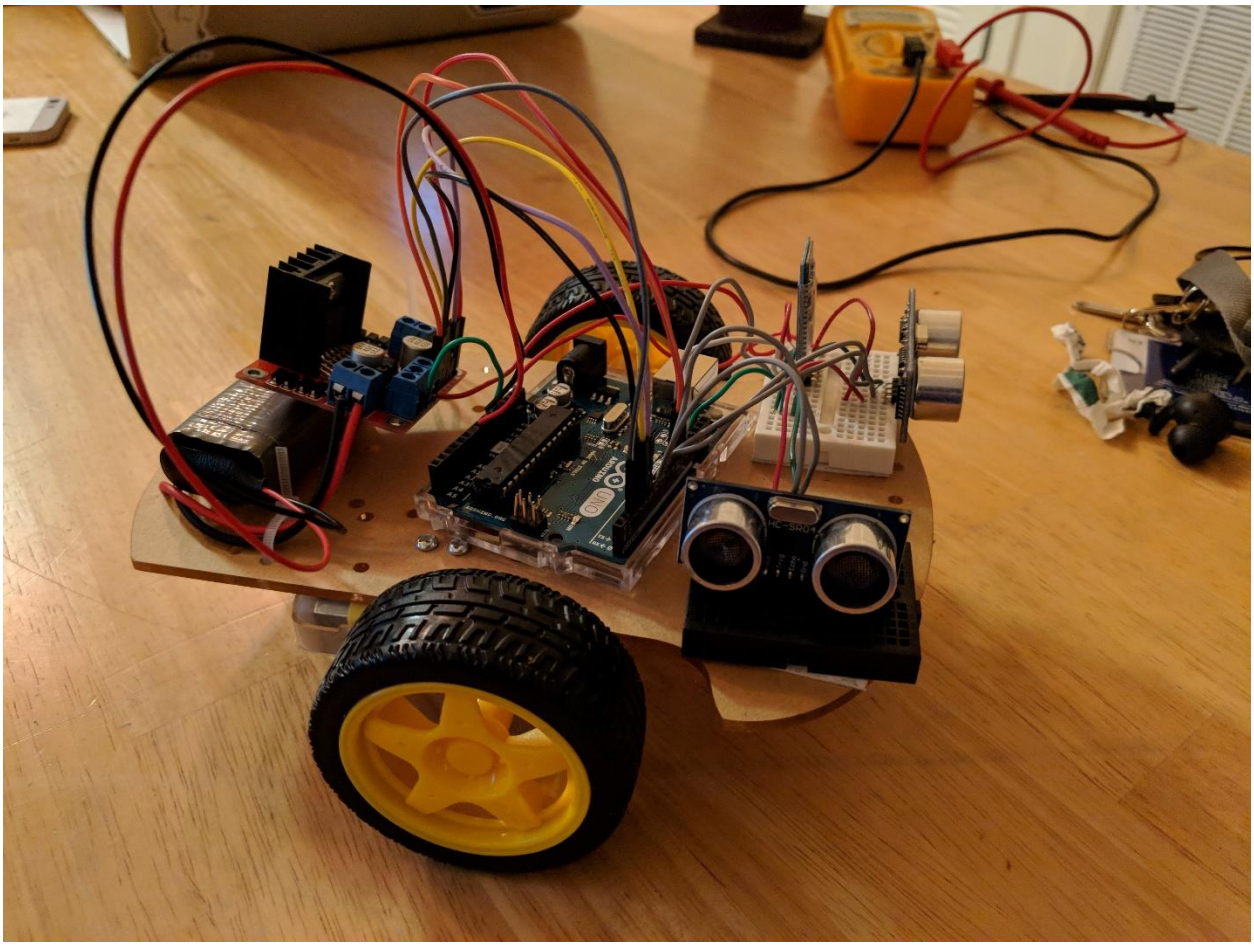
Diagram on next page.

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017



Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

Build Photograph



Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

Arduino Code

```
#include <SoftwareSerial.h>

// Constants
const double MIN_DIST = 50;
const double MAX_VEL = 200;

// Ping sensors pins
const int triggerPinLeft = 8;
const int echoPinLeft = 9;
const int triggerPinRight = 6;
const int echoPinRight = 7;

// Wheel pins
const int dir1PinA = 4;
const int dir2PinA = 5;
const int speedPinA = 10;

const int dir1PinB = 2;
const int dir2PinB = 3;
const int speedPinB = 11;

// Bluetooth pins
const int rxPin = 13;
const int txPin = 12;
SoftwareSerial bluetooth(rxPin, txPin);

// Robot state
enum RobotState {
  Idling,
  Automatic,
  Manual
};
RobotState state = Idling;

// Robot direction
enum Direction {
  None,
  Forward,
  Right,
  Backward,
  Left
};
Direction dir = Direction::None;

// Wheel speed/direction
int leftWheelDir1 = LOW;
int leftWheelDir2 = LOW;
double leftWheelSpeed = MAX_VEL;

int rightWheelDir1 = LOW;
int rightWheelDir2 = LOW;
double rightWheelSpeed = MAX_VEL;
```

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

```
void setup() {
  // Define output for direction pins
  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(speedPinA, OUTPUT);
  analogWrite(speedPinA, leftWheelSpeed);

  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);
  pinMode(speedPinB, OUTPUT);
  analogWrite(speedPinB, rightWheelSpeed);

  // Define I/O for left and right ping sensors
  pinMode(triggerPinLeft, OUTPUT);
  pinMode(echoPinLeft, INPUT);
  pinMode(triggerPinRight, OUTPUT);
  pinMode(echoPinRight, INPUT);

  // Open bluetooth pins
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);

  // Set the data rate for SoftwareSerial port
  bluetooth.begin(9600);

  // Print introduction to bluetooth device
  bluetooth.println("Robot Car Started.");
}

int var = 1;
void loop() {
  readInput();

  // Update robot car's position based on its current state
  switch (state) {
    case Idling:
      break;
    case Automatic:
      CheckObstacles();
      MoveRobot();
      break;
    case Manual:
      MoveRobot();
      break;
    default:
      break;
  }
}
```

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

```
void readInput() {
  if (bluetooth.available() > 0) {
    char cmd = bluetooth.read();

    switch (cmd) {
      case '0':
        state = RobotState::Idling;
        Stop();
        MoveRobot();
        bluetooth.println("Robot car is idling.");
        break;
      case '1':
        state = RobotState::Automatic;
        bluetooth.println("Robot car is automatically moving");
        break;
      case '2':
        state = RobotState::Manual;
        bluetooth.println("Robot car is manually moving");
        break;
      case 'w':
        if (state == RobotState::Manual) {
          MoveForward();
        }
        else {
          bluetooth.println("Error: Not in Manual mode!");
        }
        break;
      case 's':
        if (state == RobotState::Manual) {
          MoveBackward();
        }
        else {
          bluetooth.println("Error: Not in Manual mode!");
        }
        break;
      case 'a':
        if (state == RobotState::Manual) {
          MoveLeft();
        }
        else {
          bluetooth.println("Error: Not in Manual mode!");
        }
        break;
      case 'd':
        if (state == RobotState::Manual) {
          MoveRight();
        }
        else {
          bluetooth.println("Error: Not in Manual mode!");
        }
        break;
      default:
        bluetooth.println("Unknown command. Options are:\n'0' Idle\n'1' Move Automatically\n'2' Move Manually");
        break;
    }
  }
}
```

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

```
void CheckObstacles() {
  long distance_left = GetDistanceOfPingSensor(triggerPinLeft, echoPinLeft);
  long distance_right = GetDistanceOfPingSensor(triggerPinRight, echoPinRight);

  if (distance_left <= MIN_DIST && distance_left >= 10) {
    if (distance_right <= MIN_DIST && distance_right >= 10) {
      MoveLeft();
    } else {
      MoveRight();
    }
  } else if (distance_right <= MIN_DIST && distance_right >= 10) {
    MoveLeft();
  } else {
    MoveForward();
  }
}
```

```
long GetDistanceOfPingSensor(int trigger, int echo)
{
  // Read leftt pin
  // Send a trigger
  digitalWrite(trigger, LOW);
  delayMicroseconds(5);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);

  // Read in the duration
  pinMode(echoPinLeft, INPUT);
  long duration = pulseIn(echo, HIGH);

  // Convert time into distance
  return (duration / 2.0) / 29.1;
}
```

```
void MoveForward() {
  if (dir == Direction::Forward) {
    return;
  }

  bluetooth.println("Forward");
  leftWheelDir1 = LOW;
  leftWheelDir2 = HIGH;

  rightWheelDir1 = LOW;
  rightWheelDir2 = HIGH;

  MoveRobot();

  dir = Direction::Forward;
}
```

```
void MoveRight() {
  if (dir == Direction::Right) {
    return;
  }

  bluetooth.println("Right");
  leftWheelDir1 = LOW;
  leftWheelDir2 = HIGH;

  rightWheelDir1 = LOW;
  rightWheelDir2 = LOW;

  MoveRobot();

  dir = Direction::Right;

  delay(1000);
  MoveForward();
}
```

```
void MoveLeft() {
  if (dir == Direction::Left) {
    return;
  }

  bluetooth.println("Left");
  leftWheelDir1 = LOW;
  leftWheelDir2 = LOW;

  rightWheelDir1 = LOW;
  rightWheelDir2 = HIGH;

  MoveRobot();

  dir = Direction::Left;

  delay(1000);
  MoveForward();
}
```

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

```
void MoveBackward() {
  if (dir == Direction::Backward) {
    return;
  }

  leftWheelDir1 = HIGH;
  leftWheelDir2 = LOW;

  rightWheelDir1 = HIGH;
  rightWheelDir2 = LOW;

  MoveRobot();

  dir = Direction::Backward;
}

void Stop() {
  leftWheelDir1 = LOW;
  leftWheelDir2 = LOW;

  rightWheelDir1 = LOW;
  rightWheelDir2 = LOW;

  dir = Direction::None;
}

void MoveRobot() {
  delay(100);

  // Send movement to robot's wheels
  digitalWrite(dir1PinA, leftWheelDir1);
  digitalWrite(dir2PinA, leftWheelDir2);

  delay(100);

  digitalWrite(dir1PinB, rightWheelDir1);
  digitalWrite(dir2PinB, rightWheelDir2);
}
```

Bhat, Ashwin
Kosarek, Matthew
EN.530.420
Fall 2017

Budget

Part (# of Units)	$ Cost	Purpose
Arduino UNO (1)	$20	Microcontroller for controlling everything
Mini Breadboards (2)	$1.16	Wiring/connecting components together
Dual H-Bridge/Motor Driver (1)	$1.81	Interfaces between UNO and motor for control of motor direction
Chassis kit – Base, motors (2), wheels (2)	$11.97	Mount for car parts and wheels, movement
Ultrasonic Distance Sensor HC-SR04 (2)	$1.96	Obstacle detection
Bluetooth Module HC-06 (1)	$8.99	Wireless communication/control
9V battery connector (1)	$0.25	Connects battery to whole system
9V battery (1)	$1.50	Powers the whole system
Wires/Jumpers	$0.50	Connect the subsystems/components
TOTAL COST	**$48.14**	TOTAL

We still had a good amount of overhead left in the budget ($25). However, to keep the cost under the budget some sacrifices were made in terms of the quality of the parts.

Production Modifications

There is still room left for modifications if this were to be produced for real-world use. It has potential as either a toy car, or if a very large budget was available this could be used for surveillance and/or scouting. One example that comes to mind, is that if a sophisticated WiFi module and superior ping sensors were used, then a program could be written for this similar setup where a room could be mapped by driving the car around a room.

In the case of some smaller modifications, better wheels would be needed for this to be produced and sold. The wheels we used where of inferior quality. Because of this, our car would sometimes veer when trying to go forward. Better motors would also help. In addition, a current amplifier or a second microcontroller would be very beneficial. This is because with the Bluetooth module, PING sensor, and 2 motors we were approaching the current draw from the Arduino UNO and often the motors would not get enough current supplied, which would decrease the stall torque and result in the motors stalling very easily. In addition, the range of our Bluetooth module was short, so a better module would let this be more useable.