Mobile Robotics Lab Report #4 Part 1:

Arduino Controlled Robot

Heidi Gwinner

Dylan Lozon  
Quinn Austin

Kettering University

College of Engineering

Department of Electrical and Computer Engineering

CE-442: Mobile Robotics

Dr. Girma Tewolde

Spring 2024

Apr 23, 2024

[**I. OBJECTIVES 2**](#_olrevvpb5c9i)

[**II. TASKS 2**](#_1fs63ooemoxr)

[Task One: Interface the Arduino with the Robot 2](#_2ggrq472pmwg)

[Task Two: Install the Arduino IDE 2](#_8lw3dihvvpza)

[Task Three: Test the Motors 2](#_ogjg20gbq1ta)

[Task Four: Navigation Programs 2](#_he3sodlydo65)

[Task Five: Ultrasonic Sensor Integration 3](#_6eyhpxeeodd4)

[Task Six: Obstacle Avoidance 3](#_95uqk749mkjy)

[**III. Source Code 4**](#_9p4dhoqqoqxq)

[A. Updated Motor Library Header File (Motor.h) 4](#_9g7y7lmgpfsb)

[B. Updated Motor Library Source File (Motor.cpp) 5](#_rpqhan22pv4o)

[C. Square Drive Pattern (SquareDrive.ino) 6](#_e0hvcibg36k2)

[D. Figure Eight Drive Pattern (FigureEightDrive.ino) 8](#_g3m464az91p)

[E. Obstacle Avoidance Drive Behavior (ObstacleAvoidance.ino) 9](#_t0uundenprkq)

# OBJECTIVES

* Use an Arduino microcontroller to interface with a simple mobile robot.
* Write programs to navigate the robot.
* Utilize an ultrasonic sensor to implement obstacle detection and avoidance.

# TASKS

## **Task One**: Interface the Arduino with the Robot

1. The robot uses an Arduino Uno to interface with 2 DRV8220 H-bridge motor driver modules, which control two motors to drive the robot. The physical action of connecting the board to the motors was completed before the beginning of the lab.

## **Task Two**: Install the Arduino IDE

1. While it is possible to control a robot using Matlab, using the Arduino IDE is much more practical. Version 1.8.19 of the IDE was installed and used to complete the lab. In addition to the IDE, the group was provided a motor control library, which we heavily modified to include more complex functionality.

## **Task Three**: Test the Motors

1. In the Arduino IDE, the first task was to define the two motors and tell the robot to go forward. To do this, high-level functions were created to define the wheel's direction and power. These functions were called in the main loop to tell the robot to go forward for a time and then stop. Once the definitions forward and backward were created, the definitions for clockwise and counterclockwise rotations were implemented.

## **Task Four**: Navigation Programs

1. The robot drives in a four-foot square with modified versions of these functions. The robot was put through several iterations to fine-tune the motor speed and time of motion. The result is a very precise square.
2. Similarly, functions for driving in a four-foot diameter circle were created and tuned through a series of iterative tests. The functions were placed in the main loop in such a way as to create the figure-eight pattern.

## **Task Five**: Ultrasonic Sensor Integration

1. The robot has an ultrasonic sensor used for range finding. Before using the sensor for more complex tasks, a basic implementation was used for testing. A new Arduino library was added, which allowed the robot to report if there was an object in front of it.

## **Task Six**: Obstacle Avoidance

1. With a functional ultrasonic sensor, adding obstacle avoidance was relatively simple. First, the group programmed the robot to turn clockwise upon encountering obstacles to fix an error where the robot would get trapped in a corner and rotate clockwise and counterclockwise without escaping. This was fixed by having the robot turn using a while loop until the ultrasonic sensor detects an opening to drive straight.

# Source Code

## Updated Motor Library Header File (Motor.h)

/\*

Motor.h - Library for driving a dc motor using the Adafruit Adafruit TB6612 motor driver board.

Link to the motor driver board:

https://learn.adafruit.com/adafruit-tb6612-h-bridge-dc-stepper-motor-driver-breakout/overview

Created by Li Dang and Girma Tewolde, October 13, 2015.

Modified by Dylan Lozon, April 19, 2024.

Released into the public domain.

\*/

#ifndef Motor\_h

#define Motor\_h

#include "Arduino.h"

class Motor {

public:

/\*\* @brief Constructor for the Motor class.

\* @param d\_pin The digital pin connected to the direction pin of the motor.

\* @param s\_pin The analog pin connected to the speed pin of the motor. \*/

Motor(int d\_pin, int s\_pin);

/\*\* @brief Get the inversion state of the motor.

\* @return Whether the motor direction is being inverted. \*/

bool getInverted();

/\*\* @brief Set the inversion state of the motor.

\* @param invert Whether the motor direction should be inverted. \*/

void setInverted(bool invert);

/\*\* @brief Sets the speed of the motor where negative values drive in reverse.

\* @param speed The speed value to set [-100, 100]. \*/

void setSpeed(int speed);

/\*\* @brief Drives the motor forward.

\* @param speed The speed value to set [0, 100]. \*/

void forward(int speed);

/\*\* @brief Drives the motor in reverse.

\* @param speed The speed value to set [0, 100]. \*/

void backward(int speed);

/\*\* @brief Stops sending power to the motor. \*/

void stop();

/\*\* @brief Provides a very small amount of power to the motor to stop it from turning. \*/

void activeStop();

private:

// The digital pin connected to the direction pin of the motor.

int dir\_pin;

// The analog pin connected to the speed pin of the motor.

int speed\_pin;

// Whether the motor direction is inverted.

bool inverted;

};

#endif

## Updated Motor Library Source File (Motor.cpp)

#include "Arduino.h"

#include "Motor.h"

Motor::Motor(int d\_pin, int s\_pin) {

pinMode(d\_pin, OUTPUT);

pinMode(s\_pin, OUTPUT);

dir\_pin = d\_pin;

speed\_pin = s\_pin;

inverted = false;

}

void Motor::setInverted(bool invert) { inverted = invert; }

bool Motor::getInverted() { return inverted; }

void Motor::setSpeed(int speed) {

// If speed is negative, drive in reverse.

bool dir = (speed <= 0 ? LOW : HIGH);

// If inverted, invert the direction.

dir = inverted ? !dir : dir;

// Drive the motor

digitalWrite(dir\_pin, dir);

analogWrite(speed\_pin, map(speed, 0, 100, 0, 255));

}

void Motor::forward(int speed) {

digitalWrite(dir\_pin, inverted ? LOW : HIGH);

analogWrite(speed\_pin, map(speed, 0, 100, 0, 255));

}

void Motor::backward(int speed) {

digitalWrite(dir\_pin, inverted ? HIGH : LOW);

analogWrite(speed\_pin, map(speed, 0, 100, 0, 255));

}

void Motor::stop() {

analogWrite(speed\_pin, 0);

}

void Motor::activeStop() {

digitalWrite(dir\_pin, !digitalRead(dir\_pin));

analogWrite(speed\_pin, 3);

}

## Square Drive Pattern (SquareDrive.ino)

#include "Arduino.h"

#include <Motor.h>

Motor motorLeft(8, 5);

Motor motorRight(12, 6);

// Turn the robot 90 degrees to its right

void turnClockwise90() {

motorLeft.setSpeed(50);

motorRight.setSpeed(-50);

delay(430);

motorLeft.activeStop();

motorRight.activeStop();

}

// Drive the robot forward 4ft

void driveForward4ft50Percent() {

motorLeft.forward(48);

motorRight.forward(50);

delay(3100);

motorLeft.activeStop();

motorRight.activeStop();

}

// Drive the robot forward 4ft

void driveForward4ft80Percent() {

motorLeft.forward(76);

motorRight.forward(80);

delay(1969);

motorLeft.activeStop();

motorRight.activeStop();

}

// Configure motors and wait a second before starting the program

void setup() {

motorLeft.setInverted(true);

delay(1000);

}

void loop() {

// Drive in a square at 50% speed

for (int i = 0; i < 4; i++) {

driveForward4ft50Percent();

delay(250);

turnClockwise90();

delay(250);

}

delay(2500); // Wait between squares

// Drive in a square at 80% speed

for (int i = 0; i < 4; i++) {

driveForward4ft80Percent();

delay(250);

turnClockwise90();

delay(250);

}

delay(2500); // Wait between squares

}

## Figure Eight Drive Pattern (FigureEightDrive.ino)

#include "Arduino.h"

#include <Motor.h>

Motor motorLeft(8, 5);

Motor motorRight(12, 6);

// Drive in a right-side circle with a 4ft diameter

void clockwiseCircle4ftDiameter() {

// Drive in a circle

motorLeft.forward(70);

motorRight.forward(54);

delay(8015);

// Brake

motorLeft.activeStop();

motorRight.activeStop();

}

// Drive in a left-side circle with a 4ft diameter

void counterClockwiseCircle4ftDiameter() {

// Drive in a circle

motorLeft.forward(50);

motorRight.forward(70);

delay(7915);

// Brake

motorLeft.activeStop();

motorRight.activeStop();

}

// Configure motors and wait a second before starting the program

void setup() {

motorLeft.setInverted(true);

delay(1000);

}

// Drive in a figure-eight pattern with 4ft diameter circles

void loop() {

clockwiseCircle4ftDiameter();

delay(1000);

counterClockwiseCircle4ftDiameter();

delay(1000);

}

## Obstacle Avoidance Drive Behavior (ObstacleAvoidance.ino)

#include "Arduino.h"

#include <Motor.h>

#include <NewPing.h>

Motor motorLeft(8, 5);

Motor motorRight(12, 6);

NewPing sonar(A2, A3, 50);

void setup() {

motorLeft.setInverted(true);

delay(1000);

}

void loop() {

int obstacleDistance = sonar.convert\_cm(sonar.ping\_median());

// If there is no obstacle within 30cm, drive forward

if (!obstacleDistance || obstacleDistance > 30) {

motorLeft.forward(49);

motorRight.forward(50);

}

else { // Turn clockwise to avoid obstacles

motorLeft.forward(50);

motorRight.backward(50);

}

delay(50);

}