ECE/CS 6780 Embedded System Design

Project Report

Hand Gesture Robot Control

30th April 2019

Sachin Boban | Mohit Kumar | Bhawna Lakshmipathy | Farhan Nasrullah

Github | **Presentation** | **Demo**

1 Objective

The purpose of this project is to implement the fundamentals of Embedded Systems learned as part of ECE/CS 6780 course. We wanted to do something electromechanica, so we decided to make a robot that can be controlled wirelessly using gestures.

2 Introduction

Robots are electro-mechanical systems that can be operated by a computer program. Robots can be autonomous or semi-autonomous. An autonomous robot is not controlled by human and acts on its own decision by sensing its environment. A semi-autonomous robot requires some level of human decision making. Robots requiring high level of precision and speed, are mostly autonomous. But there are applications that are semi-autonomous i.e., they require some human intervention to some extent.

A gesture controlled robot is nothing but a simple vehicle whose movement is controlled by the gestures or movement of a simple controller. Depending on the gestures made using the controller, the robot can move forward, backward, left, right or come to a halt. The robot can also move forward or backward while making a left or right turn. Gesture based control of robots can be useful for disabled and/or amputated people, for remote surgery, gaming etc.

We wanted to explore another microcontroller apart from the STM32F0 and chose Arduino since it is massively popular amongst hobbyists and professionals alike. Our code contains both low level C implementation of peripheral controls we did in our course along with Arduino's high level AVR libraries.

3 Functional Block Diagram

On a high level, the project consists of 2 main parts:

1. Gesture Controller (Transmitter), which the user controls to manipulate the robot movement
2. Vehicle Controller (Receiver), which sits in the robot/vehicle and translates gestures to motion

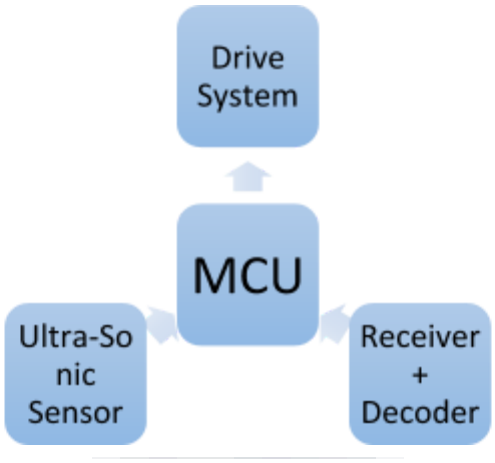
3.1 Gesture Controller (Transmitter)

The functional block diagram of the gesture controller is as given below:

Accelerometer gives the X and Y axis values of the controller. This information is then encoded by the encoder IC and transmitted to the vehicle using an RF transmitter in 434 MHz frequency. We use antennas for both the transmitter and receiver sides. The antenna is a simple 17 cm long ⅛ gauge straight copper wire.

3.2 Vehicle Controller (Receiver)

The functional block diagram of the vehicle controller is as follows:



The RF signals once received from the transmitter are decoded. These decoded values are compared with a reference (rest) value of the accelerometer and the appropriate motion is determined by the MCU and used to control the vehicle through the drive system, which is the code for motion control using the 4 DC motors. The ultrasonic sensor is used to determine presence of any obstacles in the front of the vehicle, which stops any active motion if there's any obstacle within 15 degrees of it's direct line of sight.

4 Hardware Description

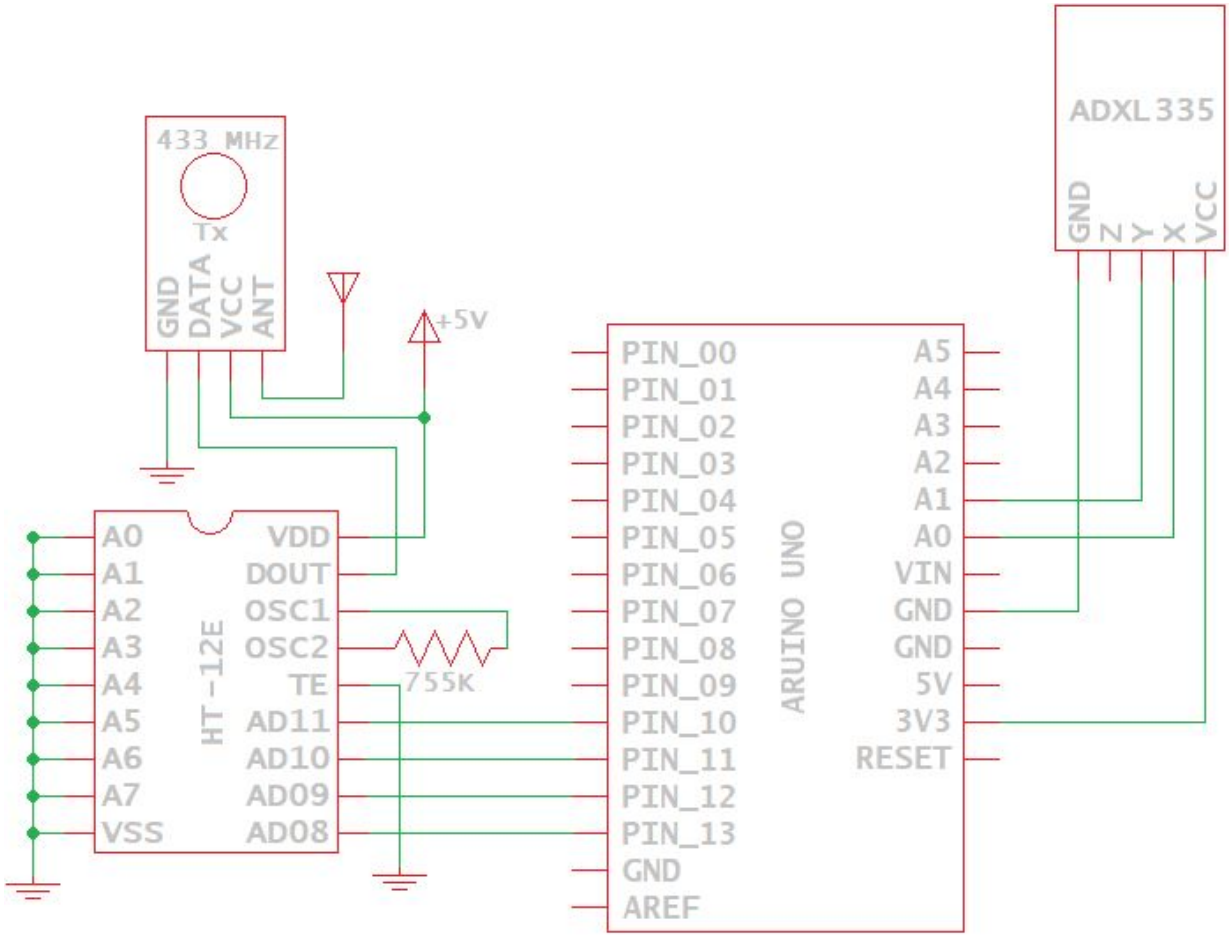
The hardware details and connectivity diagrams are provided for each of the Gesture Controller and Vehicle Controller.

4.1 Gesture Controller

4.1.1 Components

1. MCU : Arduino Uno R3
2. Accelerometer : ADXL335
3. Encoder : HT-12E
4. Transmitter : RF434 transmitter
5. Breadboard

4.1.2 Connectivity Diagram



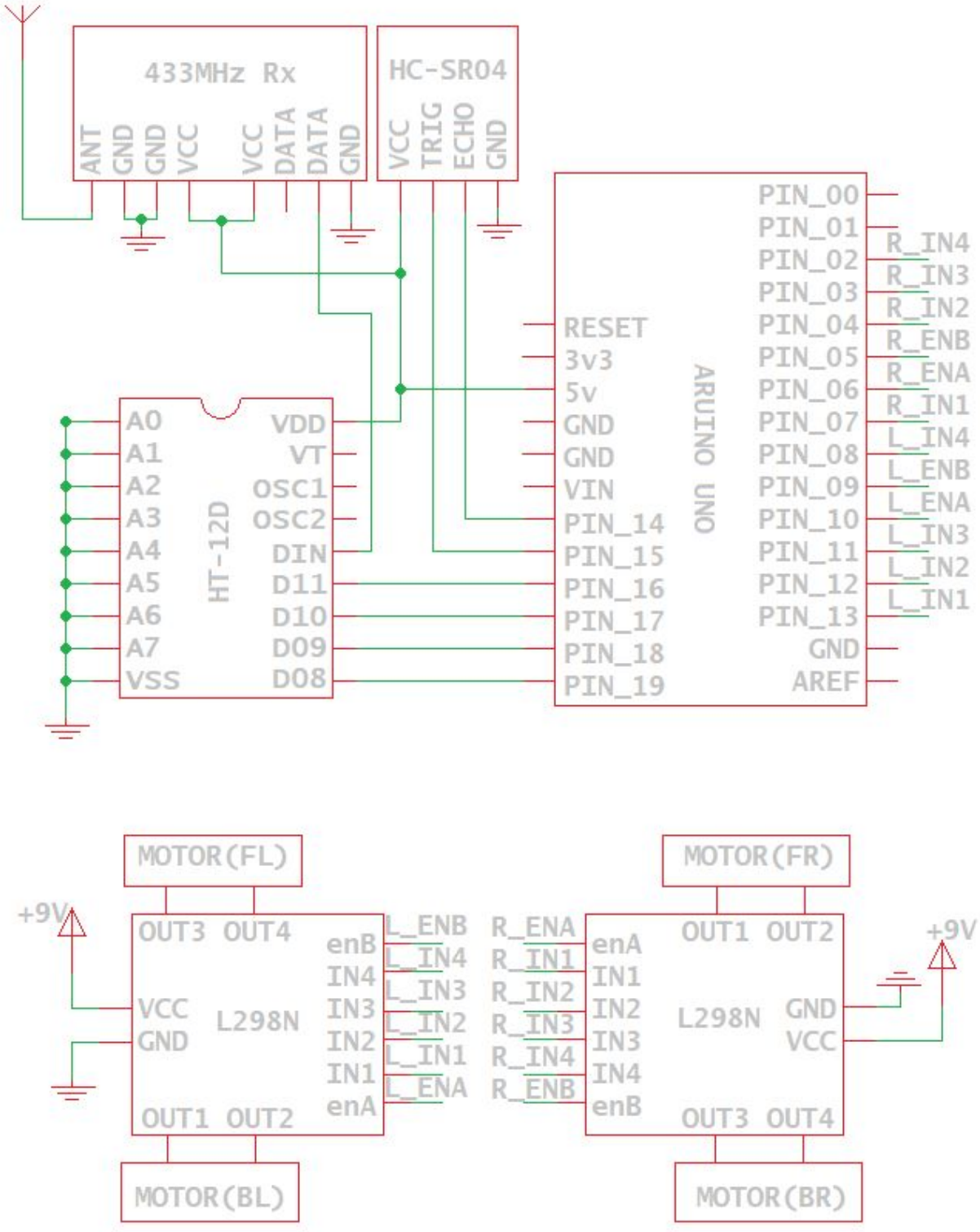
We do not use the Z axis values from accelerometer since we are operating the robot in a 2D plane. The analog values from the accelerometer and connected to the A0 and A1 analog pins and output through digital pins 10, 11, 12 and 13 of the Arduino. The encoder is always in transmission mode by setting the TE pin to zero as the pin is active low, and transmits the encoded digital data through DOUT pin.

4.2 Vehicle Controller

4.2.1 Components

1. MCU : Arduino Uno R3
2. Decoder : HT-12D
3. Receiver : RF434 receiver
4. Sensor : HC-SR04
5. H-bridge : 2 Dual H-Bridge (L298N)
6. 4 DC motors (3-12V)

4.2.2 Connectivity Diagram



5 Software

5.1 Source Code

Code for the project can be found in our GitHub page.

5.2 Gesture Controller

The X and Y axis coordinate values are read from the Accelerometer (ADXL335) through ADC interface. Depending on the coordinate values we determine the movement information to the transmitted.

Directions of movement is determined as the deviation of the X and Y values from their reference values:

- X-axis (Right/Left)
    - +ve deviation: Turn right
    - -ve deviation: Turn left
- Y-axis (Forward/Backward)
    - +ve deviation: Move forward
    - -ve deviation: Move backward

A deviation on both the axes conveys a combination of left/right turning along with moving forward/backward. This forward/backward motion while turning left/right is called as sub-direction. The absolute value of the deviation along the Y-axis determines the gear/speed of the vehicle in the forward/backward movement.

Once the direction, speed and sub-directions are determined, we configure the encoder pins as given below. Similarly, the Encoder takes 4 pins input. The 4 pins of the Encoder are configured as such:

- [Enc1 : Enc0]: Direction

Enc1	**Enc0**	**Direction**
0	0	Forward
0	1	Backward
1	0	Left
1	1	Right

- [Enc3 : Enc2]: These pins are configured depending on the direction
    - If direction is forward/backward, these pins represent the gear/speed with which the vehicle shall move forward/backward respectively
    - If direction is left/right, these pins denotes sub-direction (i.e., move forward/backward while turning):
        - [Enc2]
            - 0: No sub-direction, i.e., pure left/right turn
            - 1: [Enc3] shall be interpreted as sub-direction
        - [Enc3]: Sub-direction (valid only if Enc2 = 1)
            - 0: Forward
            - 1: Backward

5.3 Vehicle Controller

The RF signals are received on the RF433 Receiver. These signals are decoded by the decoder (HT-12D). MCU reads the output from the decoder, and interprets the pins to get the direction, gear and sub-direction as mentioned before. The vehicle also detects any obstruction in the forward direction using the ultrasonic Sensor (HC-SR04). If this distance is less than a safe distance, the vehicle is prevented from going forward. Depending on the direction, gear and sub-direction, the drive system moves the vehicle forward/backward, turns left/right, or even turn left/right while moving forward/backward (sub-direction).

Depending on direction, gear and sub-direction information, the Drive system drives the speed and direction of each of the 4 motors (using L298N) as given below:

Movement info		**Motor**							
Direction	Sub-direction	Front-Left		Front-Right		Back-Left		Back-Right	
		Direction	Gear	Direction	Gear	Direction	Gear	Direction	Gear
Forward	-	CLK	gear	CLK	gear	CLK	gear	CLK	gear
Backward	-	ACLK	gear	ACLK	gear	ACLK	gear	ACLK	gear
Left^	-	ACLK	GEAR_1	CLK	GEAR_1	ACLK	GEAR_1	CLK	GEAR_1
Left	Forward	CLK	GEAR_1	CLK	GEAR_2	CLK	GEAR_1	CLK	GEAR_2
Left	Backward	ACLK	GEAR_1	ACLK	GEAR_2	ACLK	GEAR_1	ACLK	GEAR_2
Right^	-	CLK	GEAR_1	ACLK	GEAR_1	CLK	GEAR_1	ACLK	GEAR_1
Right	Forward	CLK	GEAR_2	CLK	GEAR_1	CLK	GEAR_2	CLK	GEAR_1
Right	Backward	ACLK	GEAR_2	ACLK	GEAR_1	ACLK	GEAR_2	ACLK	GEAR_1

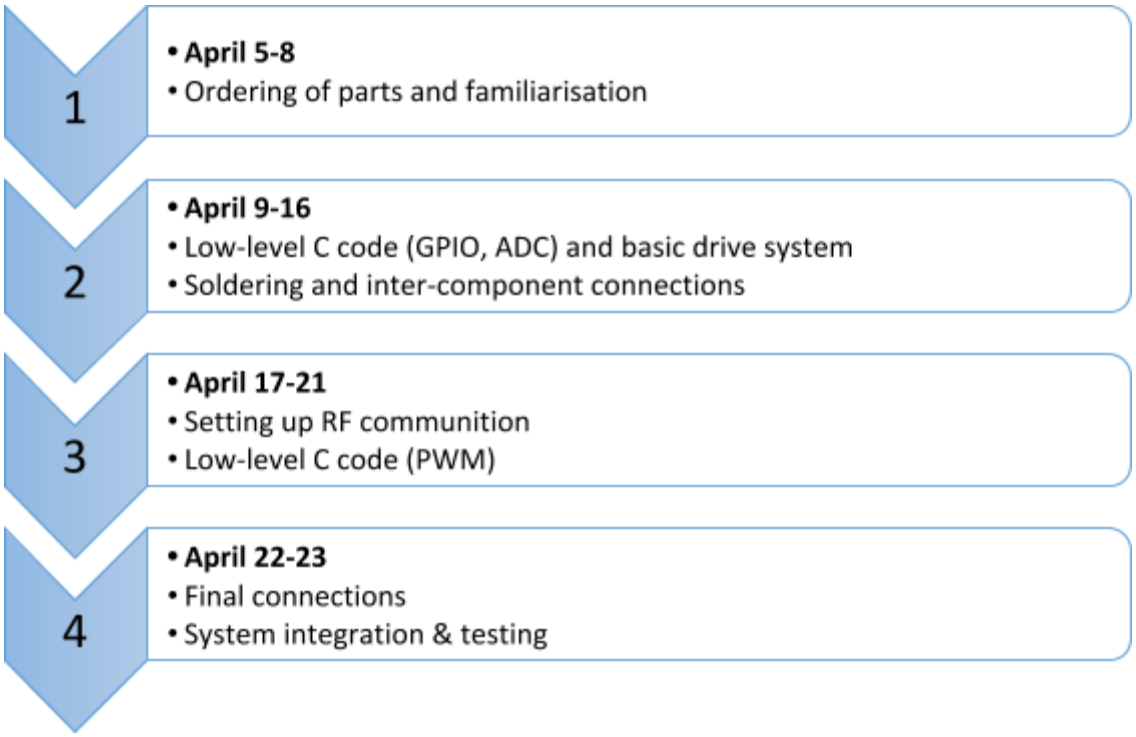
*gear: gear as received from the controller.

CLK: Clockwise
ACLK: Anti-Clockwise
^ Denotes a pure left/right turn.

The drive system supports 4 gears, with values ranging from 0 to 255:

Gear	**Speed**
GEAR_0	0
GEAR_1	150
GEAR_2	200
GEAR_3	255

6 Project Milestones



7 References

Video Demo

Youtube video

Project Presentation and Demo

Google slides presentation

Datasheet

1. ATMEGA328 Data Sheet (MCU)
2. L298N Data Sheet (Dual Full-Bridge Driver)
3. Encoder (HT-12E) Data Sheet
4. Decoder (HT-12D) Data Sheet
5. Transmitter Data Sheet
6. Receiver Data Sheet
7. Accelerometer (ADXL335) Data Sheet
8. Ultrasonic Sensor (HC-SR04) Data Sheet