ACCELEROMETER CONTROLLED CAR

Raveel Tejani
Final Project
April 2023
Phys 319

Abstract

This project shows the implementation of an accelerometer as a remote controller for a car. The design consists of the MSP430 microcontroller as the main hub for reading data from the accelerometer. The data is then transmitted through wireless communication to the car. A 434MHz wireless transmitter/receiver is used with an encoder/decoder for communication. The car is equipped with a motor regulator to control power and direction for smooth movement in all directions.

The implementation shows solid control of the cars motion based on the orientation of the accelerometer. However, average power deliverance not achievable. Motors were either fully powered or off. The wireless communication was not able to deliver pulse width modulation accurately, thus average power deliverance was abandoned for only directional commands. The results still lead to reliable and accurate control of the car with low latency. To accurately implement lower speeds, the wireless communication system would have to be changed for newer technology.

Introduction

Wireless communication is part of our daily life, from the mobile devices we carry in our backpacks and pockets, to the remotes and keys we use to control our televisions and vehicles. Wireless communication allows us transfer information at great distances, as well as control other devices. We have long range communication suited for peer-to-peer communication, while short range is mostly used for control systems. In this project, we explore the use of short-range wireless communication to control a toy car.

The accelerometer will be our form of control. it's also a device that we use routinely, as it is found in our smartphones. In smartphones, it is used for simple tasks such as correcting the orientation of the display or for more complex tasks such as control in gaming where the level of tilt would come into play. It is used in fitness trackers to monitor physical activity as well as sleep patterns. It's used in vehicles for ABS (anti-lock braking systems) as well as timing the release of airbags. We will learn about how an accelerometer works and find it serves as a great form of control that is intuitive.

Theory

In this section we discuss the theory behind all the devices that are used, specifically their inner workings at a high-level as well as how they are integrated into the project.

Accelerometer

An accelerometer is a device that allows you to measure acceleration varying on a certain axis. Our device, the 3-axis accelerometer, is essentially a system of three accelerometers that make three acceleration measurements, one for each plane of 3D space (x, y, z). There are various ways 3-axis accelerometers are made. Ours is specifically the MEMS (micro-electro-mechanical system) inertial sensor. This device uses a mass spring system paired with a capacitor. When a force is exerted on the mass, we expect the spring to extend/contract with a certain displacement. While one end of the spring is attached to the mass, the other is attached to a capacitor. This displacement causes the change in distance between the capacitor plates which then corresponds to a change in capacitance. This ultimately outputs varying voltages. From this, we have a linear relation with the earth's gravitational force, which we can use to determine the orientation of the accelerometer in accordance with any of the three axes.

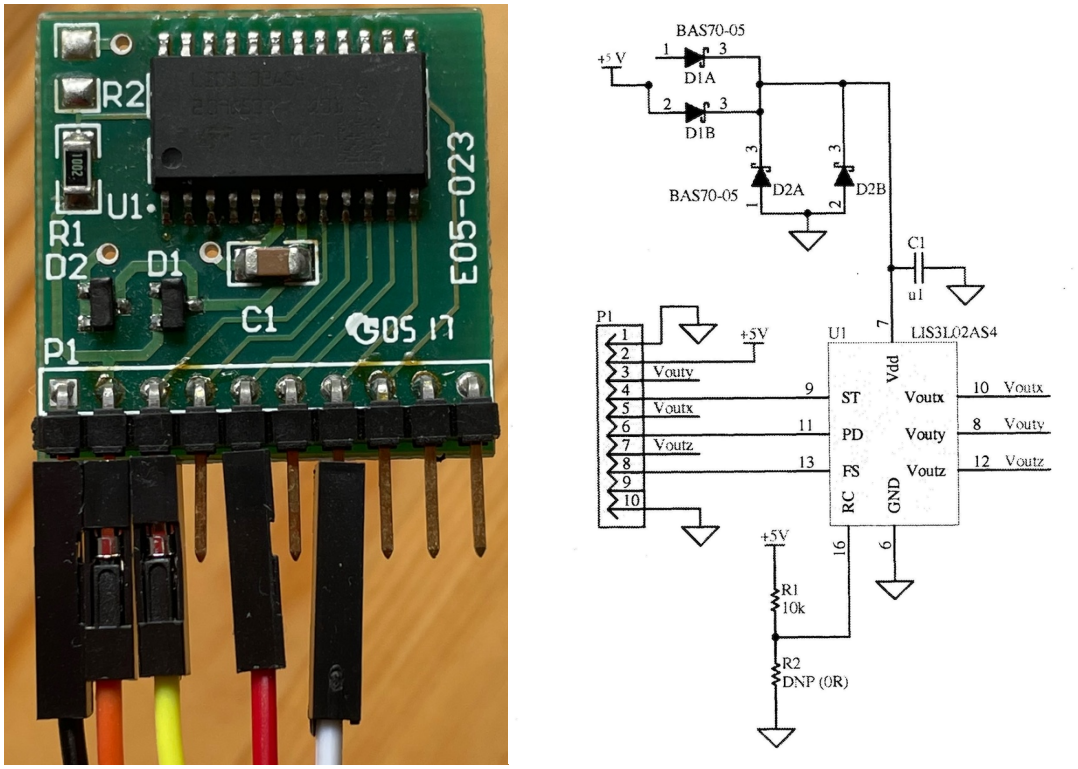


Figure 1. the physical accelerometer (LIS3L02AS4) used in the project, shown attached to a board with their specific pins indicated on the diagram to the right. To explain the relevant pins (where we see cables attached) in the physical image, from left to right, we have ground, 5V, voltage from the y-axis, voltage from the x-axis and voltage from the z-axis.[2]

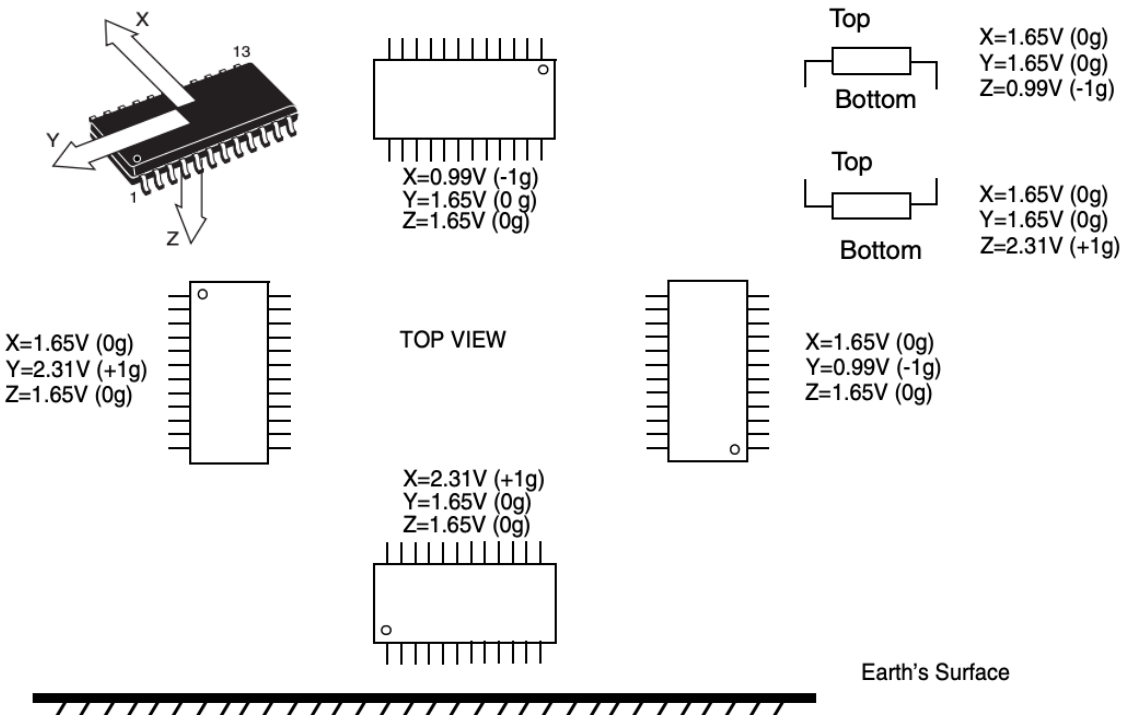
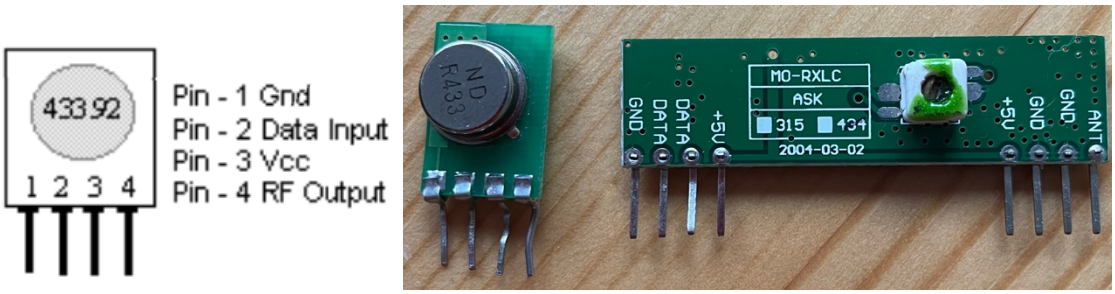


Figure 2. The accelerometer (LIS3L02AS4) output voltages are shown based on orientation of the device against the earth's surface.[2]

For our purposes, we will be using the x-direction to control forward and reverse motion of the car, while the y-direction will be used for turning. The z-direction will not be used.

Wireless Transmitter / Receiver

The wireless transmitter / receiver is a pair of devices that allows for wireless communication. The device pair we use is operating on the 434 MHz frequency with amplitude modulation (AM). To elaborate, the transmitter device generates a wave at the frequency of 434 MHz called a carrier wave. To transmit information, the amplitude of this carrier wave is altered to encode the information before being transmitted. On the receiver end, the signal is picked up, removes the original carrier wave, leaving behind the original transmitted information.



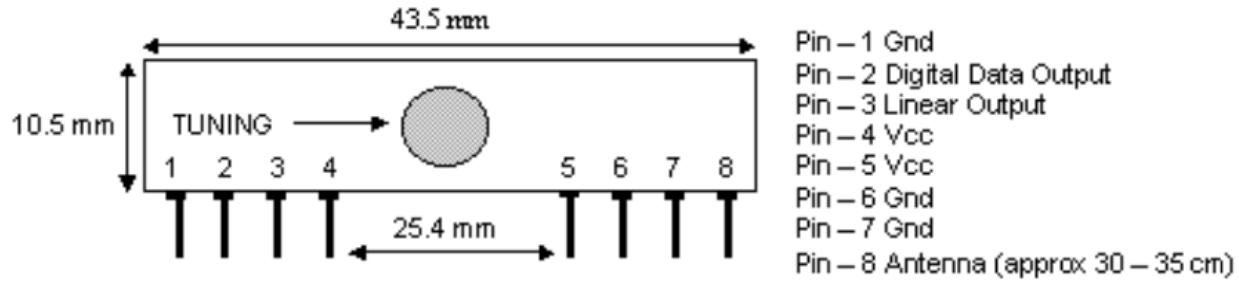


Figure 3. The wireless transmitter/receiver pair used in this project. In the physical image of the devices, the device on the left-hand side is the transmitter (TWS-434) corresponding to the left schematic, and the device on the right-hand side is the receiver (RWS-434), corresponding to the bottom schematic.[3]

8-bit Encoder / Decoder

The 8-bit encoder is a useful device for encoding 8 bits of information into one encoded line that is then decoded into the same 8 bits of information through the decoder. This is useful in many cases for example in data compression as well as data communication where the latter is used in this project. These devices are paired with the wireless communication we mentioned above to encode data into the 434MHz frequency wave.

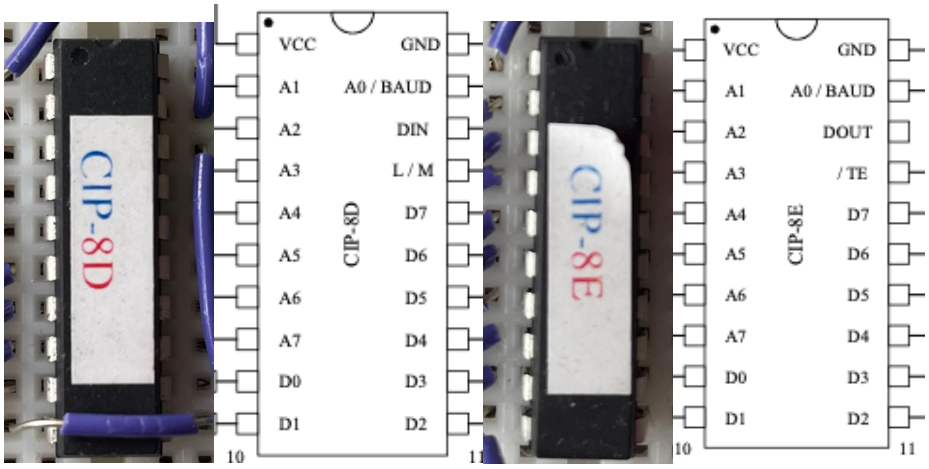


Figure 4. The encoder (CIP-8E) and decoder (CIP-8D) used in this project can be seen in the image along with their corresponding schematic.[4]

Let's discuss the various pins on these devices. Pins A0 to A7 are used to establish a unique address relationship between the encoder and decoder.[4] They need to be high or low and cannot be left floating. We will ground all these pins on both devices. D0-D7 are the data pins and since we only need to transmit four bits of information, we can ground the other four. Arbitrarily choosing D4-D7 as the data pins, the remaining D0-D3 will be grounded. We will input our data

through D4-D7 on the encoder side and retrieve the same data on D4-D7 on the decoder side. We have the DOUT/DIN pins that output or receive the encoded line from the transmitter/receiver respectively. These are 5V devices, thus we will input 5V at VCC, and we have our GND pins for ground. A 0.1uF capacitor is recommended between VCC and GND to reduce any noise. The /TE (Transmit Enable) pin found on the encoder needs to be grounded, for continuous transmission. Lastly, we have a L/M pin on the decoder. This is setting to a "latch or momentary mode", depending on whether the pin is set to high or low respectively. The latch mode would output the last received 8-bit binary data value until new information is received. Momentary mode works on a timer, after 65.5 milliseconds, the decoder would time out and send all the data lines to zero. For our case, momentary mode (achieved by setting the pin to low) is ideal, as we are using these pins to control motors. If all the pins are set to zero after the timer, the motors will stop. This is useful in a scenario where the car is out of range for transmission, we would want the car to stop and not continue at its last trajectory.

Motor Regulator

The motor regulator is a device used to control the power and direction of the motors. This will be used for the car's motors. We are using the L298N motor driver device, which doubles as a voltage regulator.

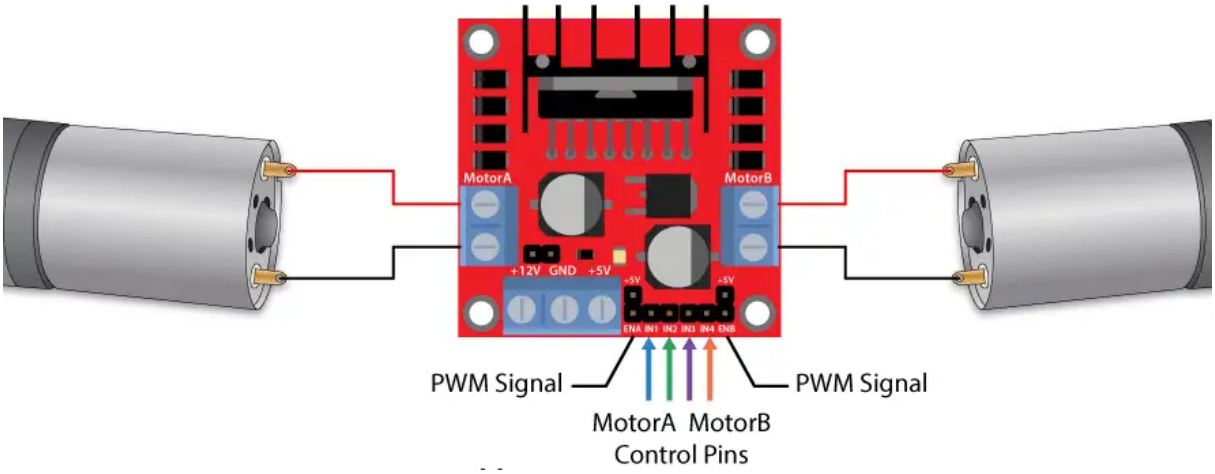


Figure 5. Image of the various lines on the motor regulator. [5]

Feeding any voltage higher than 5V on the 12V line, will allow for the voltage regulator to kick in and supply 5V out through the 5V line. Input pins 1 and 2 correspond to motor A and pins 3 and 4 correspond to motor B. The direction of the motor is controlled by H-bridges based on our inputs.

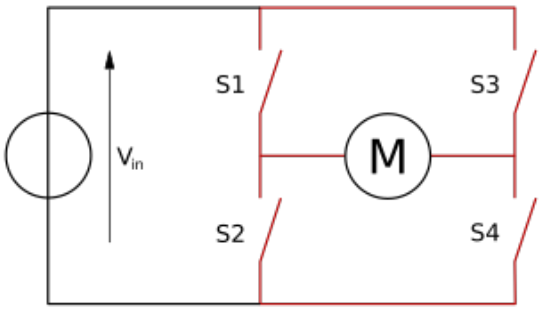


Figure 6. schematic of an H-bridge. [6]

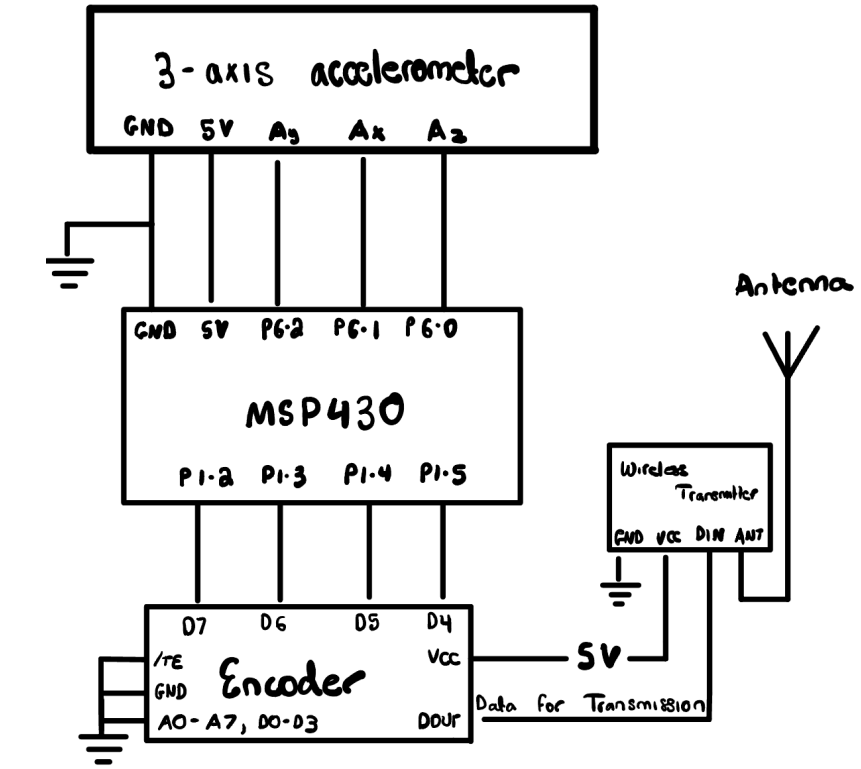
We can describe how are inputs work with the schematic of the H-bridge above. Let's see how we would control motor A. Assume that S1 and S4 are associated with input 1, and S2 and S3 are associated with input 2. From this we can gather the following table:

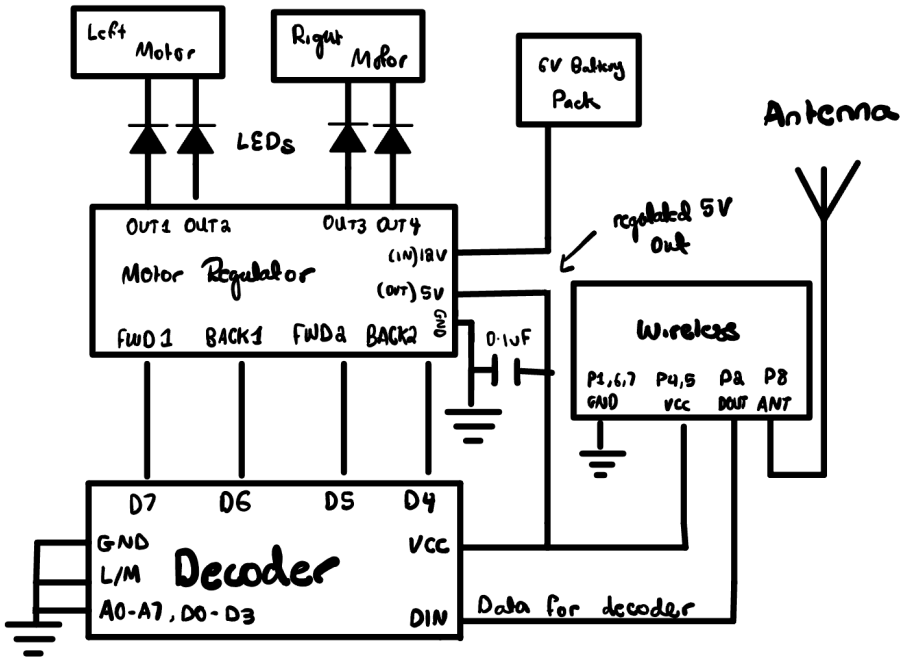
Input 1	Input 2	S1	S2	S3	S4	Motor
0	0	OFF	OFF	OFF	OFF	OFF
1	0	ON	OFF	OFF	ON	Spinning
0	1	OFF	ON	ON	OFF	Spinning (Opposite)
1	1	ON	ON	ON	ON	OFF

We can see our choice of simple choice of input can specify direction. This works the same way for motor B but with Input 3 and 4. One other useful feature is this motor regulator can take PWM signals, so we can control average power to motors with duty cycles.

Apparatus

The main pair of circuits is shown below, where the top schematic is the transmission board along with the microcontroller, and the bottom schematic is the receiver board found on the car.

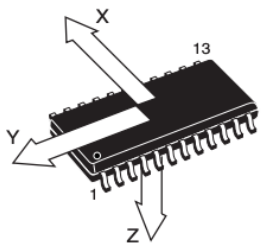




Starting from the transmission board, we have the accelerometer. The microcontroller uses multi-channel analog to digital conversion to collect the data on P6.0 - P6.2. This data is put through a series of conditions (discussed in the next few paragraphs) which then triggers P1.2 through P1.5 high or low. This data is passed off to the encoder through pins D4 – D7. This is then encoded in the 434MHz carrier wave and transmitted through to the receiver board. The encoded data runs through the decoder, the original information from P1.2 - P1.5 is retrieved and passed off to the motor regulator to control the power and direction delivered to the motors.

As stated previously, the acceleration in the x-direction is used to control forward and backward motion, while the acceleration in the y-direction is used to control turning. The z-direction is not used.

The values from the ADC ranges from 2000 to 4095, depending on the orientation of the device. the value in the x-direction change based on a tilt around the y-axis. While the value in the y-direction changes based on rotation around the x-axis. We can conceptualize this with the image on the right. These values correspond to a voltage which ultimately corresponds to a “g” (gravitational force) value where the lowest value 2000 is -1g while 4095 would correspond to +1g.



To control forward or backward motion, we can first find the middle value which we can gather by either testing the device or realizing the range is a linear relation. This value is between 3000 - 3050. We can construct a simple if statement to control the direction of the motors, as seen below:

x-acceleration	P1.2	P1.3	P1.4	P1.5	Motor A	Motor B
x>3200	0	1	0	1	BWD	BWD
x<2800	1	0	1	0	FWD	FWD
2800<x<3200	0	0	0	0	OFF	OFF

The last condition is a buffer, to keep the motors from jittering back and forth and implements a brake.

Turning can be constructed in two ways, having only one motor running, or having two motors running in opposite directions. The goal was to make the turning a smooth transition. The values of the y-acceleration were put through a larger if statement to have slow turning where only one motor is on (a small tilt of the accelerometer), and fast turning where both motors are on and in opposite directions (a large tilt of the accelerometer). Which motors are turned off or switch direction is also dependent on whether the whole vehicle was moving forward or backward. To implement this, one if statement is nested within each of the first two x-acceleration conditions above. The last condition, where the motors are kept off, no turning was implemented.

We can see below the nested if function when x-acceleration exceeds 3200 (where both motors would be running backwards if no turning was involved):

y-acceleration	P1.2	P1.3	P1.4	P1.5	Motor A	Motor B
3200 < y < 3300	0	0	0	1	OFF	BWD
y > 3300	1	0	0	1	FWD	BWD
2650< y < 2850	0	1	0	0	BWD	OFF
Y < 2650	0	1	1	0	BWD	FWD

We have the same middle value for the y-acceleration so we can create a buffer where nothing happens when the y-acceleration is in between 2850 and 3200. Starting from the top line, if it slightly exceeds 3200 (small tilt), motor A is turned off. If it exceeds 3300 (large tilt), motor A is turned on in the forward direction. Similarly, we control motor B in the final two lines, first turning off motor B, then running it forward for sharper turning.

This is implemented similarly when the x-acceleration is less than 2800 (forward movement), where we have a motor is first turned off in a small tilt then set to run backwards in a large tilt:

y-acceleration	P1.2	P1.3	P1.4	P1.5	Motor A	Motor B
3200 < y < 3300	0	0	1	0	OFF	FWD
y > 3300	0	1	1	0	BWD	FWD
2650< y < 2850	1	0	0	0	FWD	OFF
Y < 2650	1	0	0	1	FWD	BWD

The link to the code can be found at the end of this report.[7]

We can now have a look at the physical implementation on the next few pages as well as all the conditions with example images.

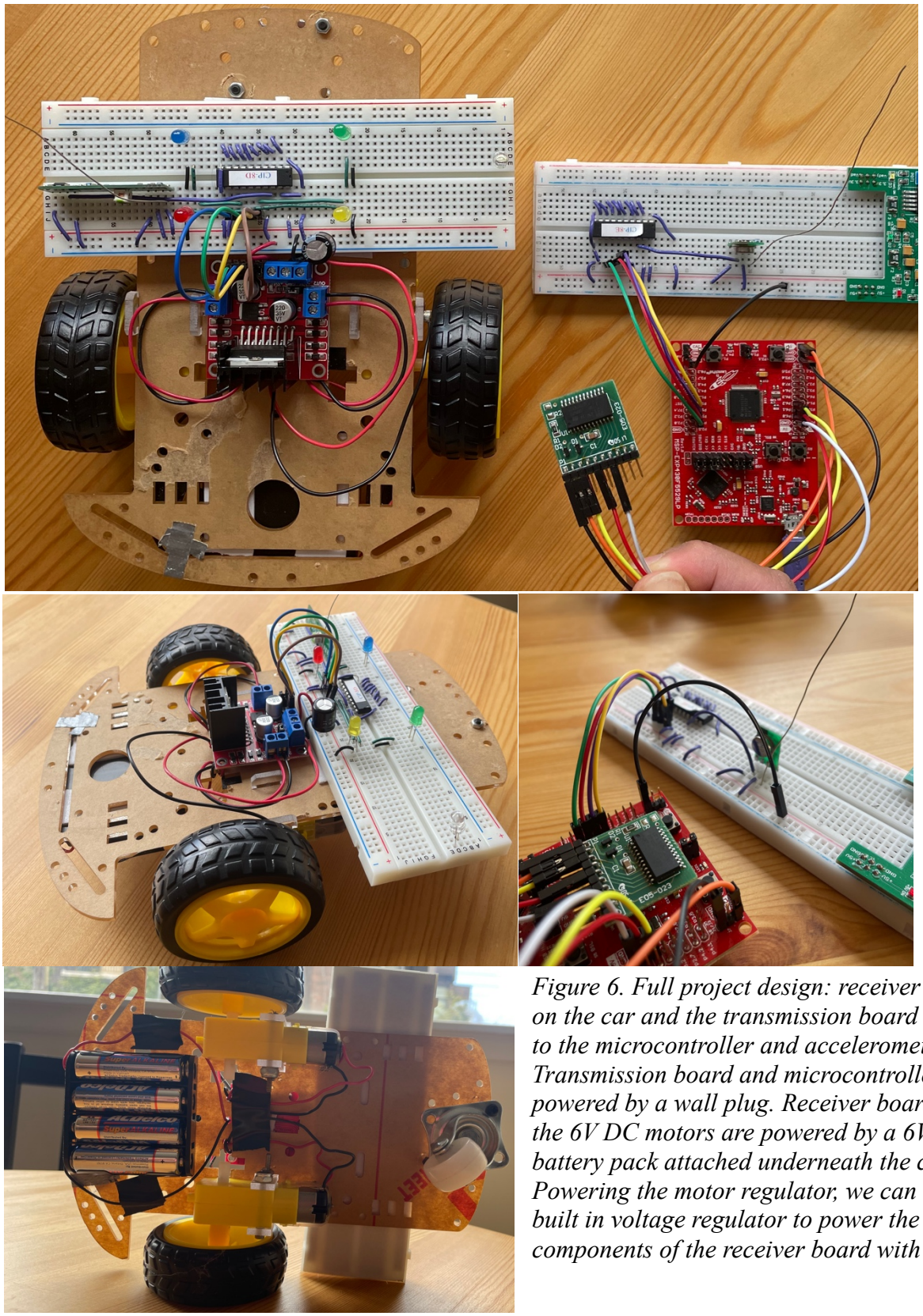
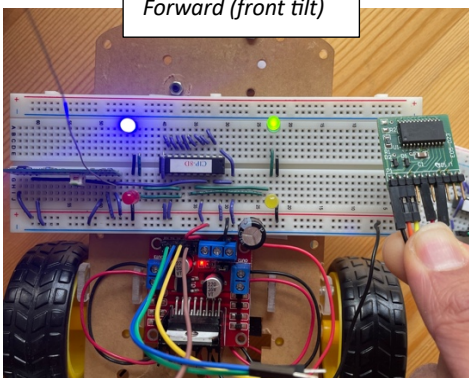


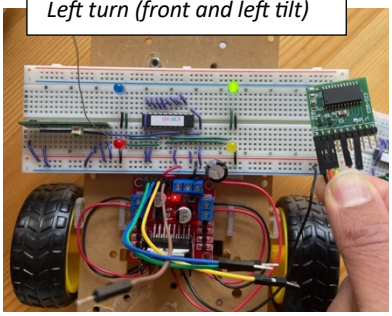
Figure 6. Full project design: receiver board on the car and the transmission board hooked to the microcontroller and accelerometer. Transmission board and microcontroller is powered by a wall plug. Receiver board and the 6V DC motors are powered by a 6V battery pack attached underneath the car. Powering the motor regulator, we can use it's built in voltage regulator to power the components of the receiver board with 5V.

The LEDs on the board proved to be a useful tool when debugging. The positions of the LEDs on the receiver board corresponds to a motor and direction. The left LEDs correspond to the left motor and similarly the right LEDs correspond to the right motor. The LEDs at the front, if turned on, signify forward direction in both motors and similarly the LEDs in the back signify reverse direction. For the example images below, the motors are disconnected, we will instead see which LEDs respond to the orientation of the accelerometer. We can see every condition sent to the car. The left column represents the first if statements, while the arrows indicate the row of nested if statements.

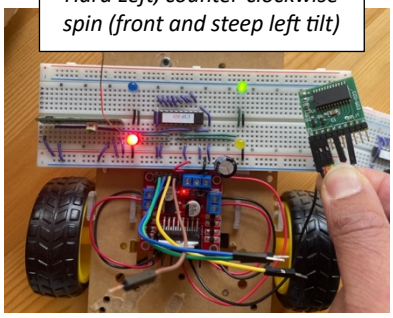
Forward (front tilt)



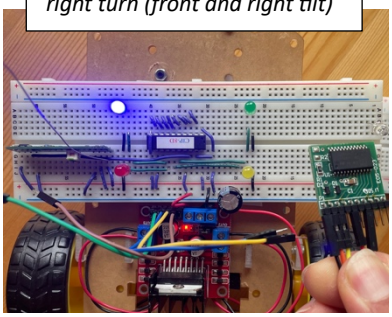
Left turn (front and left tilt)



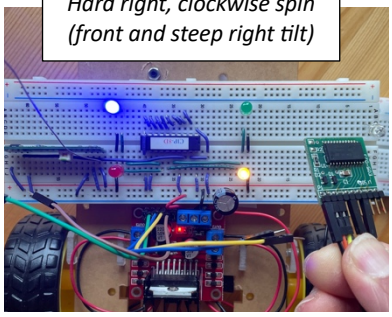
Hard Left, counter-clockwise spin (front and steep left tilt)



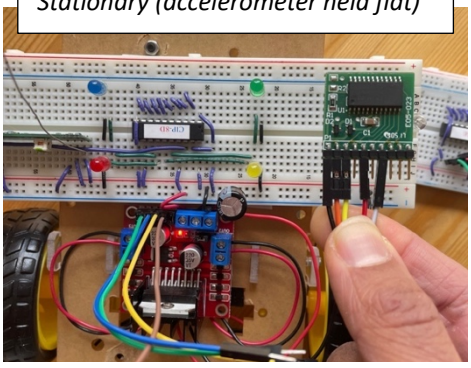
right turn (front and right tilt)



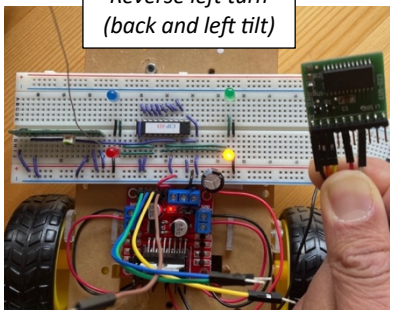
Hard right, clockwise spin (front and steep right tilt)



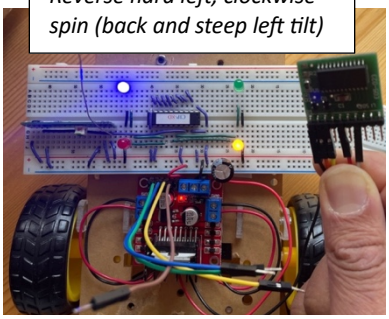
Stationary (accelerometer held flat)



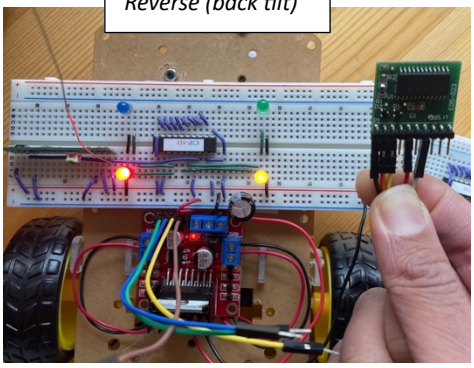
Reverse left turn (back and left tilt)



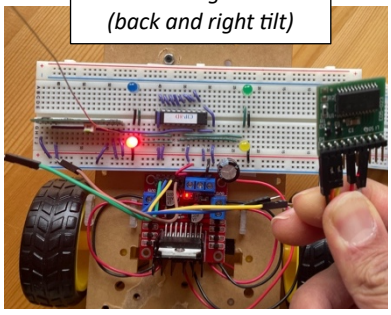
Reverse hard left, clockwise spin (back and steep left tilt)



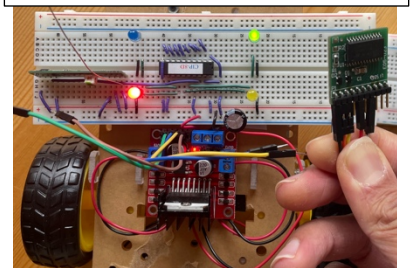
Reverse (back tilt)



Reverse right turn (back and right tilt)



Reverse hard right, counter-clockwise spin (back and steep right tilt)



Results and Discussion

The implementation as seen on presentation day showed solid control of the cars motion based on the orientation of the accelerometer. the device was very responsive, and the range spanned across the room.

An objective of this project was to implement slower speeds based on the degree of tilt of the accelerometer. However, this objective was abandoned as the wireless communication technology used in this project is old. The idea was to use pulse width modulation for average power. Upon initial testing, it was found that the transmission line and receiving line did not have equal periods. If given a PWM with a period on the order of a hundred microseconds, the receiving line delivered accurate transmission but not on the same period. The receiver end seems to be on the order of a hundred milliseconds. Upon further investigation its possible this is due to the 10ms guard time inserted between each encoded packet in the decoder.[4] To qualitatively explain this, if we attempted to transmit a pulse width modulation with a 50% duty cycle, an LED powered directly on the transmission line would appear 50% dimmer, which we know is a result of average power deliverance. However, on the receiving line, An LED would appear to be blinking. It appeared to be the correct duty cycle (tested a few different percentages) but over a longer period. To visibly see the LED turning on and off tells us the average power deliverance to the motors would not be smooth. And as expected, with some quick testing it produced a “jerky” motion and not a good representation of a slower speed. However, our implementation and results still lead to reliable and accurate control of the car with low latency.

To improve this device, I would likely rework the wireless transmission with newer technology to improve transmission speeds and thus power deliverance. I would also upgrade the DC 6V motors used for the car. These are not the fastest motors and the motor regulator on the board can support up to 12V. Housing for the accelerometer would be important as well as my current implementation and demonstration involved holding on to the cables attached to the accelerometer. A significant improvement would be to have the accelerometer powered with a smaller microcontroller and battery that could fit on the palm of your hand. This would allow for movement with the accelerometer since you are restricted to power from the wall plug.

Conclusion

Overall, the construction of the accelerometer-controlled car was an enjoyable endeavor. This project primarily focused on understanding the various hardware used. Putting all the pieces together, provided an invaluable learning experience not typically faced in a physics degree. Learning about accelerometers and wireless communication gives you insight into the complex nature of these devices that are so seamlessly intertwined in our daily life. This likely goes for many devices we use routinely. In projects like this, problem-solving and time management skills play a key role as well. And these skills were further developed during the last six weeks, where we faced challenges and adapted around problems. Lastly and mainly, I'm happy the project was feasible in six weeks and the results although not as initially planned, were quite successful.

References

1. Accelerometer Board
   https://phas.ubc.ca/~kotlicki/Physics_319/accelerometer_board.jpg

2. Accelerometer Data Sheet
   https://datasheetspdf.com/pdf-file/918998/STMicroelectronics/LIS3L02AS4TR/1

3. Wireless Transmitter/Receiver
   https://datasheetspdf.com/pdf-file/635003/ETC/TWS-434/1

4. CIP-8E Encoder / CIP-8D Decoder
   https://phas.ubc.ca/~kotlicki/Physics_319/CIP-8.pdf

5. L298N Motor Driver
   https://howtomechatronics.com/wp-content/uploads/2017/08/L298N-Block-Diagram-Current-Flow-How-It-Works.png

   https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

6. H-bridge
   https://en.wikipedia.org/wiki/H-bridge#/media/File:H_bridge.svg

7. The code (a copy can be found on the next page)
   https://github.com/raveelt/electronics-lab/blob/main/final-project/main.c

```c
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{

  ADC12CTL0 = ADC12SHT02 + ADC12ON + ADC12MSC;             // Sampling time, ADC12 on, automatic multiple conversions
  ADC12CTL1 = ADC12SHP | ADC12CONSEQ_1 | ADC12CSTARTADD_0;// sampling timer, multichannel, starting memory address
  ADC12MCTL0 = ADC12INCH_0;                               //selects A0 to be stored in memory ADC12MEM0
  ADC12MCTL1 = ADC12INCH_1;
  ADC12MCTL2 = ADC12INCH_2+ADC12EOS;        //selects A2 to be stored in memory ADC12MEM2 and this memory to be the
last of sequence
  ADC12CTL0 |= ADC12ENC;                    // ADC enable conversions
  P6SEL |= 0b00000111;                      // allow ADC on pin 6.0,6.1,6.2
  P1DIR = 0b00111100;                       //configuring P1.2-1.5 for motor control


  while (1)
  {
    ADC12CTL0 |= ADC12SC;                   // Start sampling


    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait


    int volts_z = ADC12MEM0; // not used
    int volts_x = ADC12MEM1; //good for forward / reverse movement
    int volts_y = ADC12MEM2; // good for turning


    if (volts_x > 3200){ //Car in reverse condition


        if(volts_y > 3200 && volts_y < 3300){ //slow turn reverse right
            P1OUT &= ~BIT3;
            P1OUT |=  BIT5;
            P1OUT &= ~BIT2;
            P1OUT &= ~BIT4;
        }
        else if(volts_y < 2850 && volts_y > 2650){ //slow turn reverse left
            P1OUT &= ~BIT5;
            P1OUT |=  BIT3;
            P1OUT &= ~BIT2;
            P1OUT &= ~BIT4;
        }

        else if(volts_y > 3300){ //fast turn reverse right - counterclockwise spin
            P1OUT &= ~BIT3;
            P1OUT |=  BIT5;
            P1OUT |=  BIT2;
            P1OUT &= ~BIT4;
        }
        else if(volts_y < 2650){ //fast turn reverse left - clockwise spin
            P1OUT &= ~BIT5;
            P1OUT |=  BIT3;
            P1OUT |=  BIT4;
            P1OUT &= ~BIT2;
        }
        else{  // car moving in reverse, no turning
        P1OUT |=  BIT3;
        P1OUT |=  BIT5;
        P1OUT &= ~BIT2;
        P1OUT &= ~BIT4;
        }



    }

    else if(volts_x < 2800){


        if(volts_y > 3200 && volts_y < 3300){ //slow turn right
            P1OUT &= ~BIT3;
            P1OUT |=  BIT4;
            P1OUT &= ~BIT2;
            P1OUT &= ~BIT5;
        }
        else if(volts_y < 2850 && volts_y > 2650){ //slow turn left
            P1OUT &= ~BIT5;
            P1OUT |=  BIT2;
            P1OUT &= ~BIT3;
            P1OUT &= ~BIT4;
```

```
        }

        else if(volts_y > 3300){ //fast turn right - clockwise spin
            P1OUT &= ~BIT2;
            P1OUT |=  BIT4;
            P1OUT |=  BIT3;
            P1OUT &= ~BIT5;
        }
        else if(volts_y < 2650){ //fast turn left - counterclockwise spin
            P1OUT &= ~BIT3;
            P1OUT |=  BIT2;
            P1OUT |=  BIT5;
            P1OUT &= ~BIT4;
        }
        else{    // car moving forward, no turning
            P1OUT |=  BIT2;
            P1OUT |=  BIT4;
            P1OUT &= ~BIT3;
            P1OUT &= ~BIT5;
        }

    }
    else{       // motors off - buffer condition

        P1OUT &= ~BIT2;
        P1OUT &= ~BIT3;
        P1OUT &= ~BIT4;
        P1OUT &= ~BIT5;

    }


  }

}
```