🚗 ECE 302 Speed Control Report

1. Information

- **Bench Number: 409**
- **Name(s)**: David Lee, Seungho (John) Lee
- **Date**: 10/3/2025

2. Statement of Objectives

---

1. To design and implement a small-scale car system with functional subsystems (Power MOSFET, voltage regulator, Hall Effect sensor).
2. Understand the interaction between electrical, mechanical, and control components.
3. To implement and maintain speed control (4 ft/s) of the car in various scenarios (level, downhill, uphill).

3. Overview of Key Subsystems and Components

- Battery: 7.2V and 9.6v
    - 7.2V is used to power the MOSFET
    - 9.6V is used to power the voltage regulator
- Power Regulation Method
    - Once the 9.6V is connected to the voltage regulator, it is outputted to two 5V regulators and one 6V regulator. One 5V regulator is connected to the hall effect board and the other (which is currently unused) will be connected to the video camera later. The 6V regulator is connected to the servo.
- Drive System:
    - The servo allows steering of the wheels using a controller.
- Control system: PSoC

    - Programmable microcontroller. Takes in inputs from the hall effect sensor to adjust the duty cycle for the pulse width modulation (pwm) which is outputted into the MOSFET.
    - Includes speed control using proportional and integral control.
- Sensing Subsystem: Hall Effect Sensor
    - Magnet → sensor off. No magnet → sensor on.
    - Using the period between each falling edge, we can derive the speed of the car.
    - Once gravity comes in effect (downhill, uphill), proportional and integral control (using the error between the current speed and desired speed) are incorporated to adjust pwm values for the motor.
- Communication:
    - Wireless remote for steering.
    - Programmed speed control into the car for adjustment through hall effect outputs.
- All components:
    - Power Distribution: slot 1,2,3,4,5,6
        - 2,3 are used for 7.2V battery where 2 is positive and 3 is ground. The large switch connects the positive terminal of 7.2V from 2 into 1 so that anything connected through 1 only turns on when this switch is on. 1 is connected to the power mosfet. 4,5 are used for 9.6 battery where 4 is ground and 5 is positive. The small switch connects the positive terminal of 9.6V from 5 to 6 so that anything connected through 6 only turns on when this switch is on. The switch is connected to the PSoC.
    - Power MOSFET: turns on the motor
        - Acts essentially as a transistor. When both switches are on, the 7.2V is connected to the source and through the motor while the 5V signal comes from the PSoC which powers on the motor. When the small switch is off, the drain is no longer connected to the source which means there is a chance that the drain has some voltage remaining. If that voltage happens to be greater than 7.2V, the flyback diode serves as a regulator and makes sure the drain side doesn't have a higher voltage than 7.2V. In addition, from the PSoC to the gate, there is a bleeder resistor and a gate resistor. Gate resistor is to prevent any current surge from damaging the PSoC. The bleeder resistor is to drain out any current so that the car is on and off as intended
    - Voltage Regulator
        - Described above
    - Hall Effect Circuit Board:
        - The resistor is used to decrease noise. The board is connected to the hall effect where it receives signal. Then, the signal tells the board to send either a 5V signal or no signal to the PSoC
    - PSoC
        - We programmed the speed control onto the PSoC. We added the initial speed using the hall effect signal and the duty cycle. Then, we adjusted speed using proportional and integral control. Derivative control was not

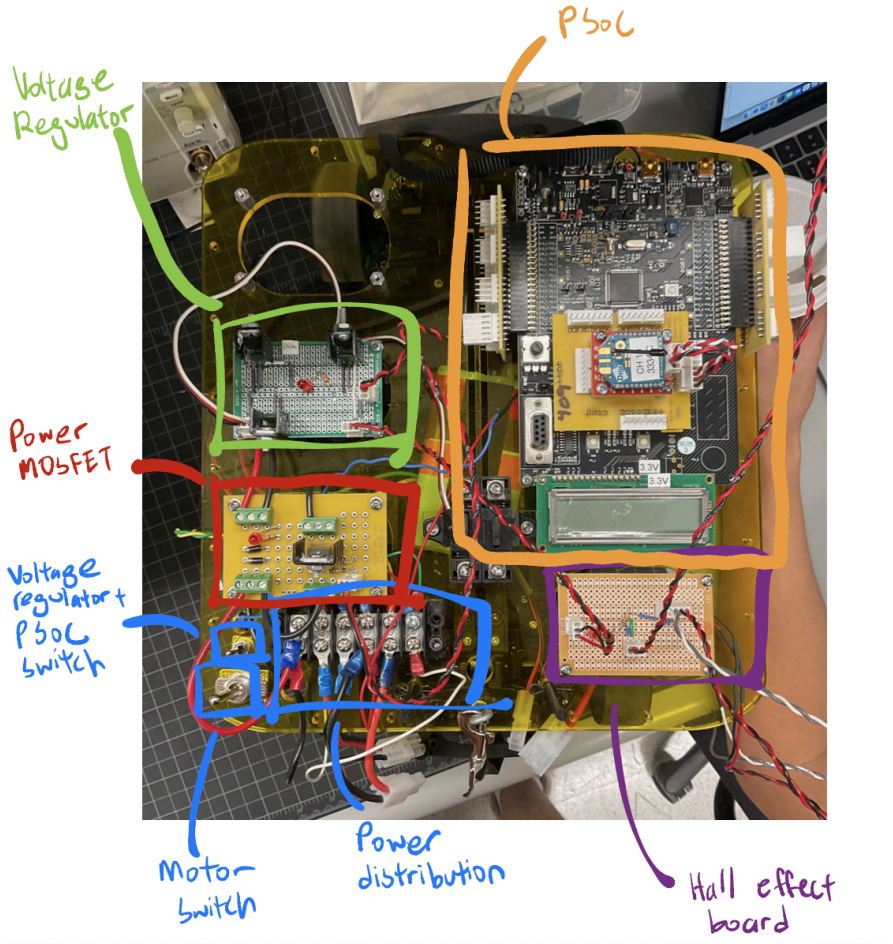
used.

4. Schematic Diagram of Circuits

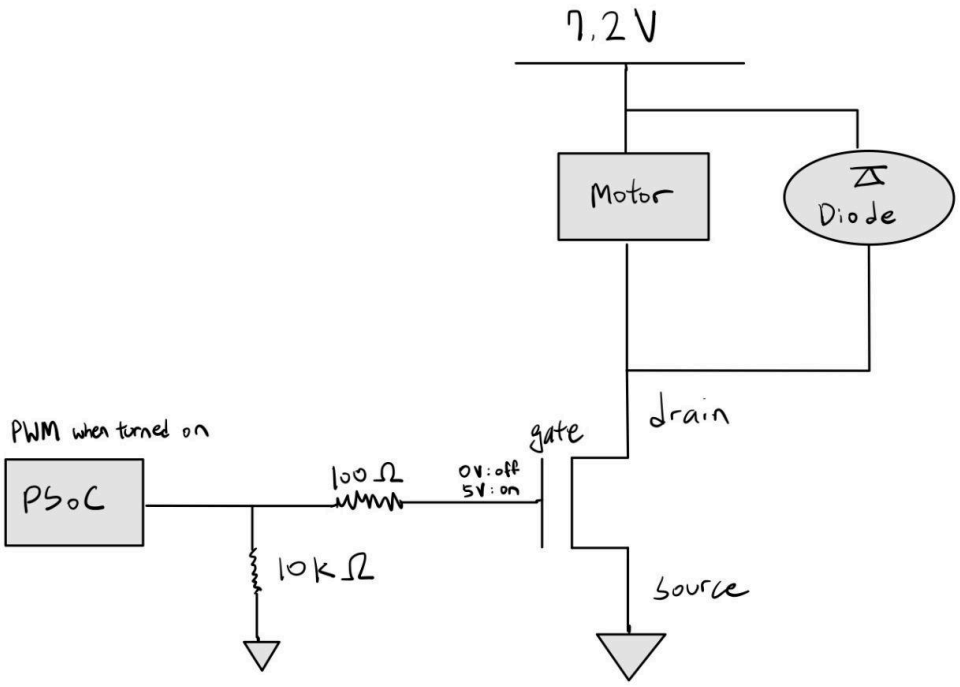
---

*Note: Circuits were hand drawn and labeled because we no longer have access to CircuitLabs

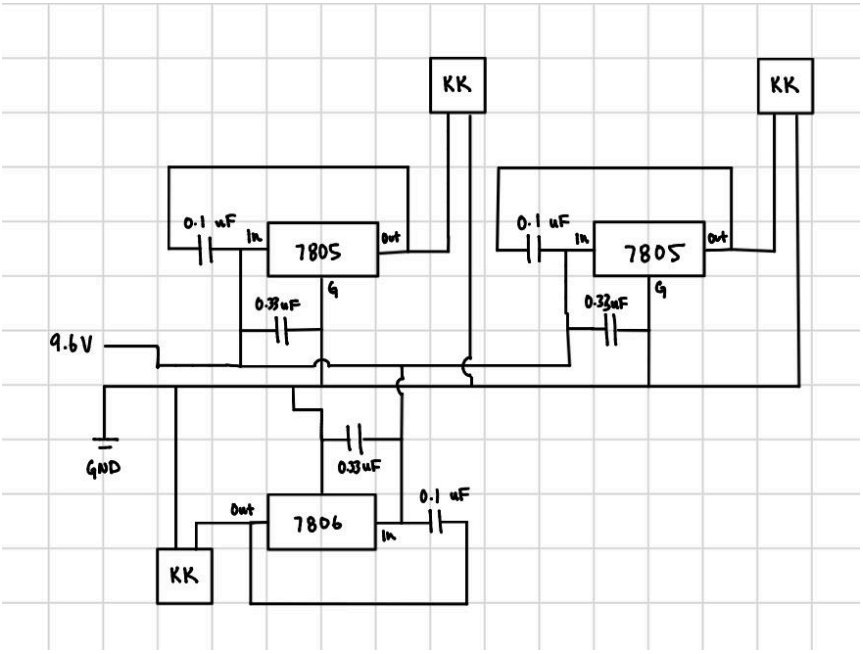
Top-down view of the car



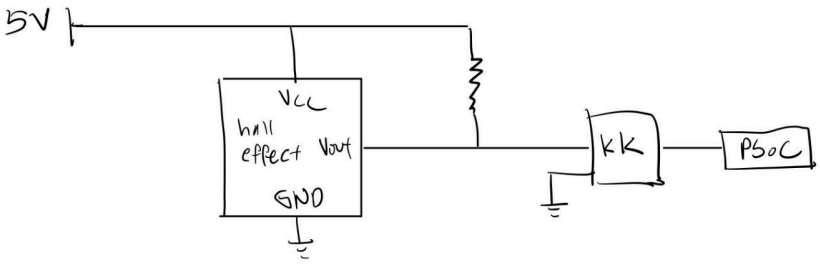
Power MOSFET



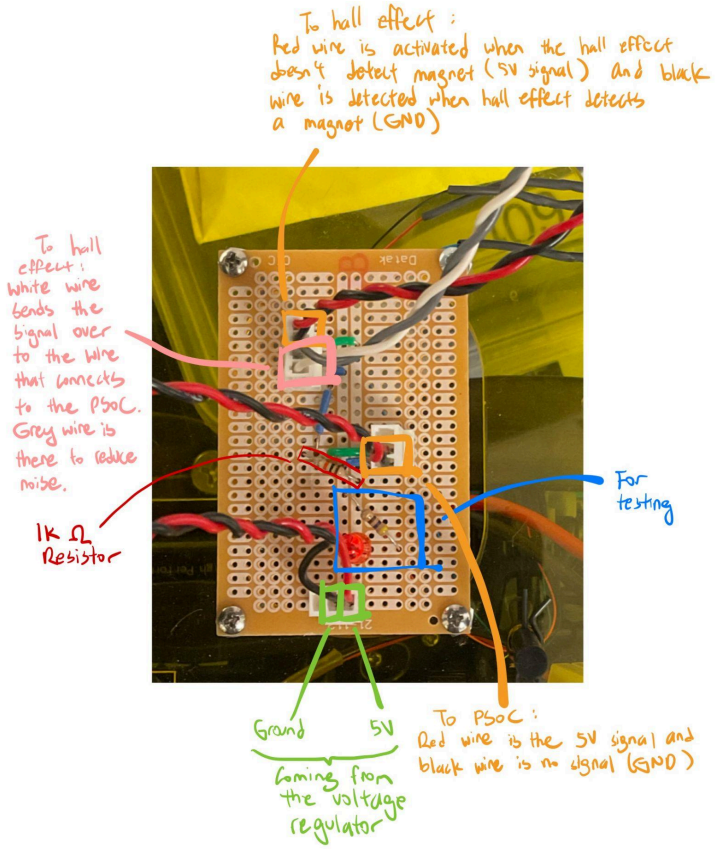
Voltage Regulator



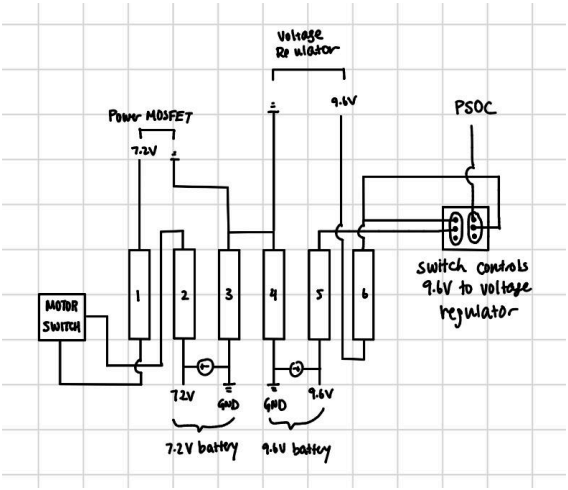
Hall Effect Circuit Board



Resistor is 1KΩ

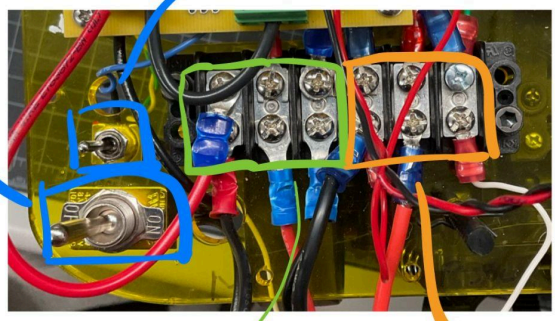


Power Distribution



Controls the 7.2 voltage into the power MOSFET

Controls the 9.6 voltage into the voltage regulator and PWM the PSoC uses to control speed



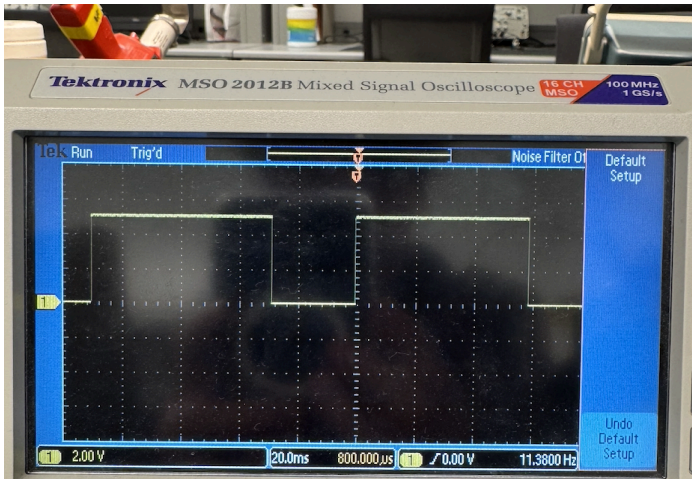
Middle is the positive terminal of 7.2V and the right is the ground. The big switch connects the positive terminal to the left and the left is connected to the power MOSFET so that the 7.2V is sent over to the power MOSFET when the switch is on.

Middle is the positive terminal of 9.6V and the left is the ground. The small switch connects the positive terminal to the right and the right is connected to the PSoC through the switch as well as the voltage regulator.

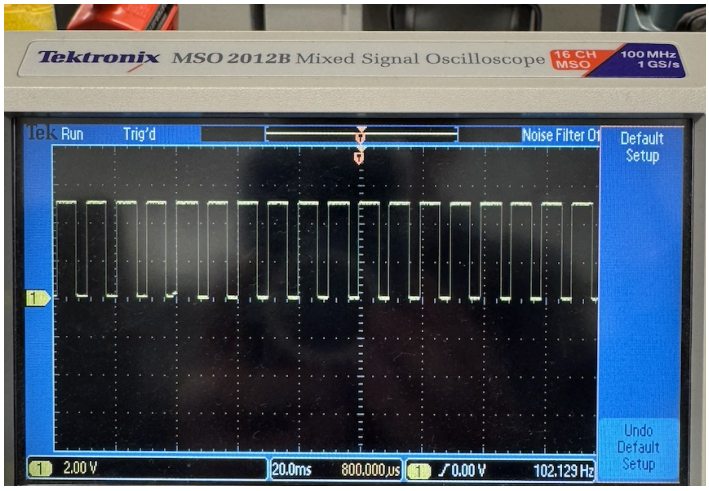
5. Data and Results

Testing Hall Effect Sensor Readings:

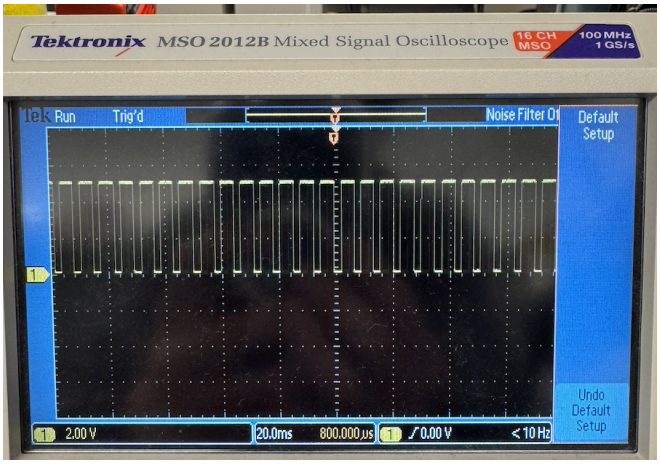
Set compare value = 10 [min 0, max 255]



Set compare value = 30



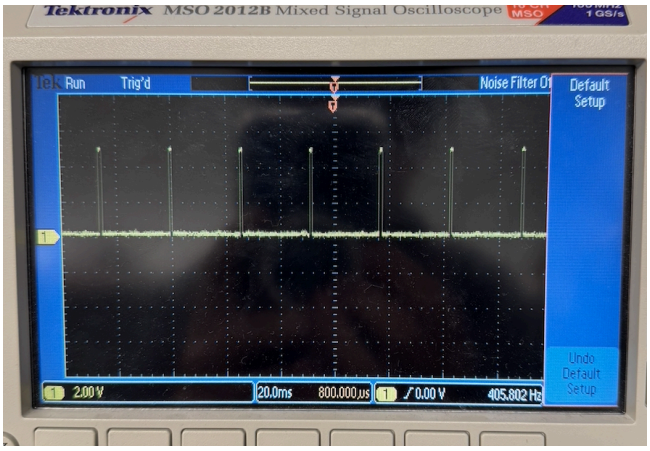
Set compare value = 50



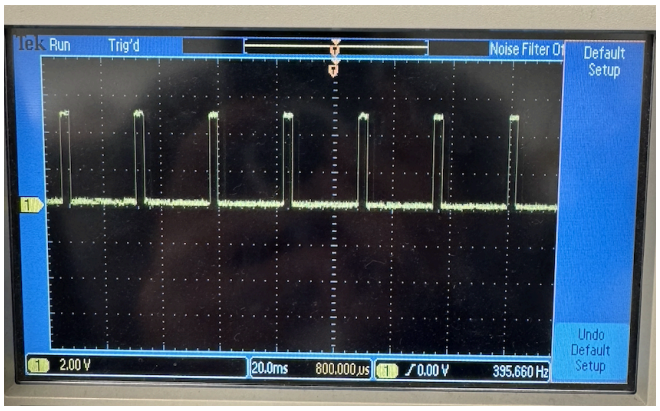
As the compare value is increased, our duty cycle increases so we should see the Hall Effect outputting a higher frequency signal since it is proportional to the rotational speed which is shown by the decrease in the period from one rising edge the next through the oscilloscope as we increase the compare value from 10 to 50.

PSOC PWM Readings:

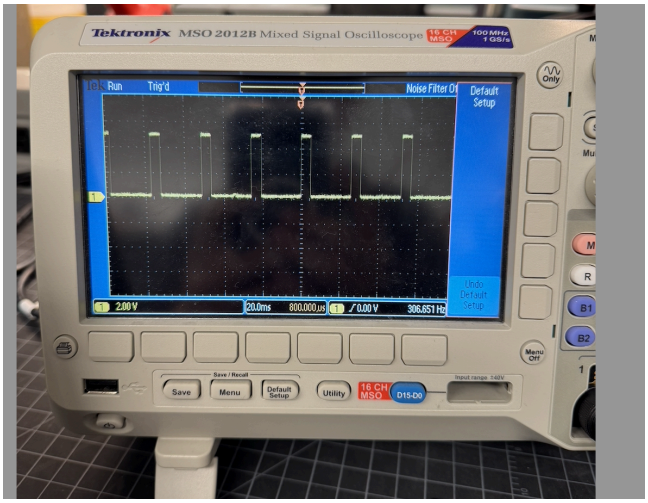
Set compare value = 10



Set compare value = 30



Set compare value = 50

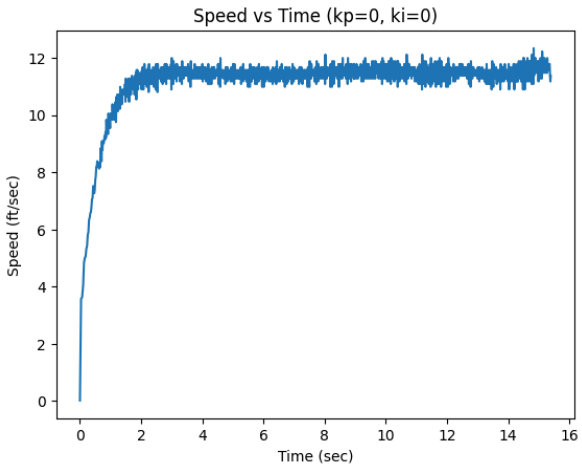


As the compare value increases, the duration that the PSOC outputs 5V to the Power MOSFET increases. As the compare value increases from 10 to 30, we see that it stays on 5V for a longer period of time. The increase from 30 to 50 is less noticeable due to scaling but there is an increase in the duration that signal stays on 5V.

Mount Testing (various K_P and K_I)

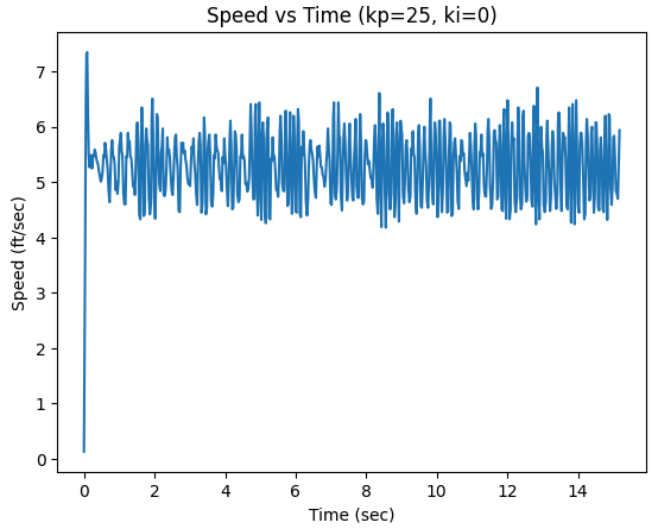
Note: Tested by setting K_P = 0, 25, 50 and K_I = 0, 5, 10 and running the car for 15 sec on the combinations made from these parameters. K_P is the proportional control constant and K_I is the integral control constant.

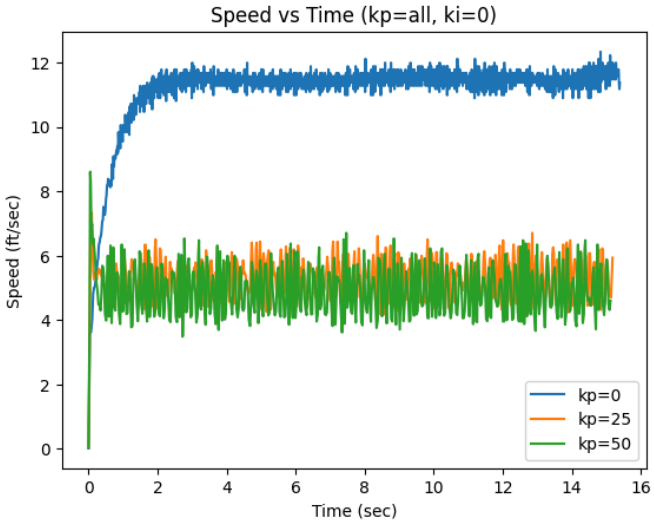
Reference K_P and K_I set to 0:



Observation: Since there is no error correction, we see that the speed increases to around 11 ft/sec and stays constant as the car is mounted.

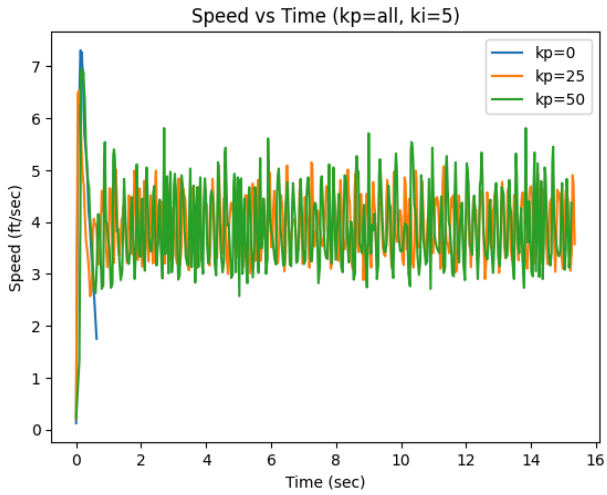
Increasing K_P:



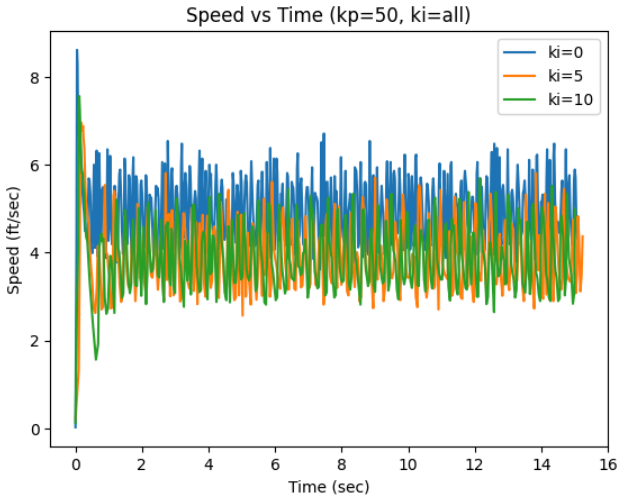


Observation: When we increase the K_P value, the speed is pushed down to be near 4 ft/sec, however steady state error persists. A higher K_P value seems to push the error closer to 0, however if you look at the next image below, there will be greater fluctuations in the error correction.

Increasing K_P with set K_I:

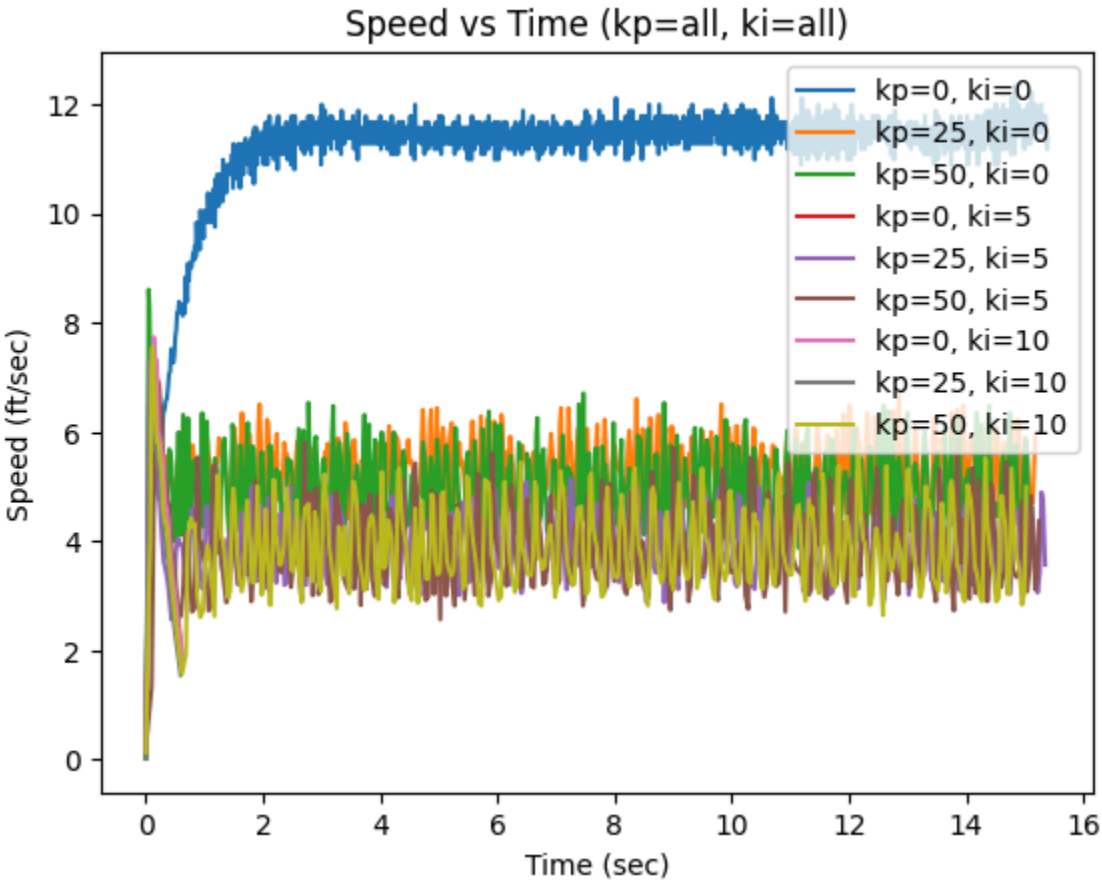


Increasing K_I with set K_P:



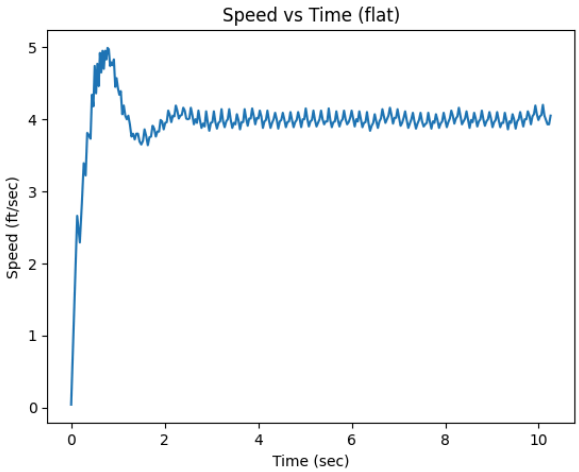
Observation: After including the integral control and increasing our K_I we see that the steady error is removed and the system stabilizes to fluctuate around 4 ft/sec. K_I = 5, 10 seem to have similar fluctuations.

Aggregate:

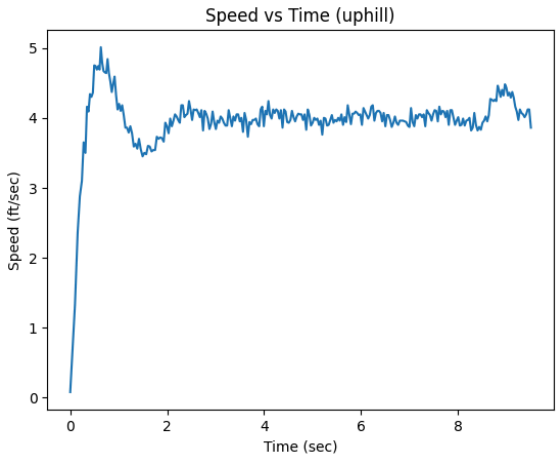


Plot Speed with K_P = 30 and K_I = 5

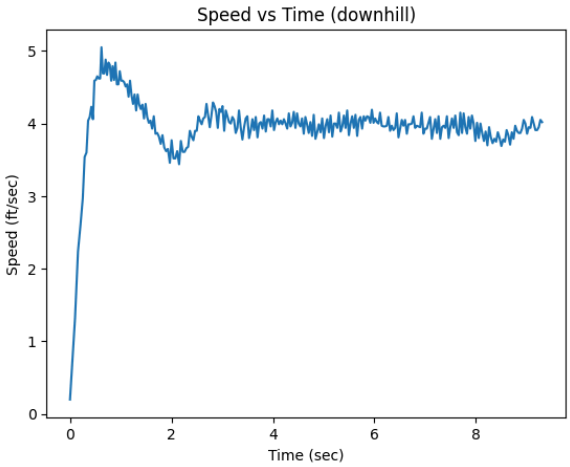
Flat:



Uphill:



Downhill:



Final Result:

Flat (s) (2% error tolerance)	Downhill (s) (10% error tolerance)	Uphill (s) (10% error tolerance)
9.99 s	9.12	9.00

We were able to hit the objective time and speed (4 ft/s) for all portions of the test within the error bounds!

6. Discussion

One of the big challenges when building this car was choosing the right MOSFET to power the motors. Initially, we had chosen a p-channel MOSFET. However, we later realized that using a n-channel MOSFET was better because it had better efficiency, higher current handling, and easier gate drive. So we had to resolder and switch to the IRF1018E MOSFET which could handle a high duty cycle for the PWM that drove the motors without overheating. We actually had to desolder twice because the initial n-channel MOSFET could not handle the high duty cycle due to its voltage threshold.

For our PID control, a challenge was choosing the right proportionality constants (K_P and K_I) to maintain speed control. If we had K_P to be a large value, we saw a really large fluctuation in speed as the car tried to speed up and slow down to hit the target speed. Initially, we wanted to check if the control was doing the right thing so we set our base PWM to be a low value, and set the K_P to be a large value to see if the car sped up to match its target speed. From there, we adjusted the K_P for the proportional control to reduce the fluctuations and then added the integral control. The challenge here was finding the right balance between K_P and K_I. While K_P adjusted the current error of the car speed to its target speed, adding the K_I, which keeps track of the accumulated error over time, helped smooth these fluctuations out which was especially helpful when we were testing the car going downhill and uphill. We found that our K_I (set to 5 in our code) does not have to be a large value and that it should be a smaller value than the K_P (set to 30 in our code) for our car since the purpose of it is to remove the steady state error.

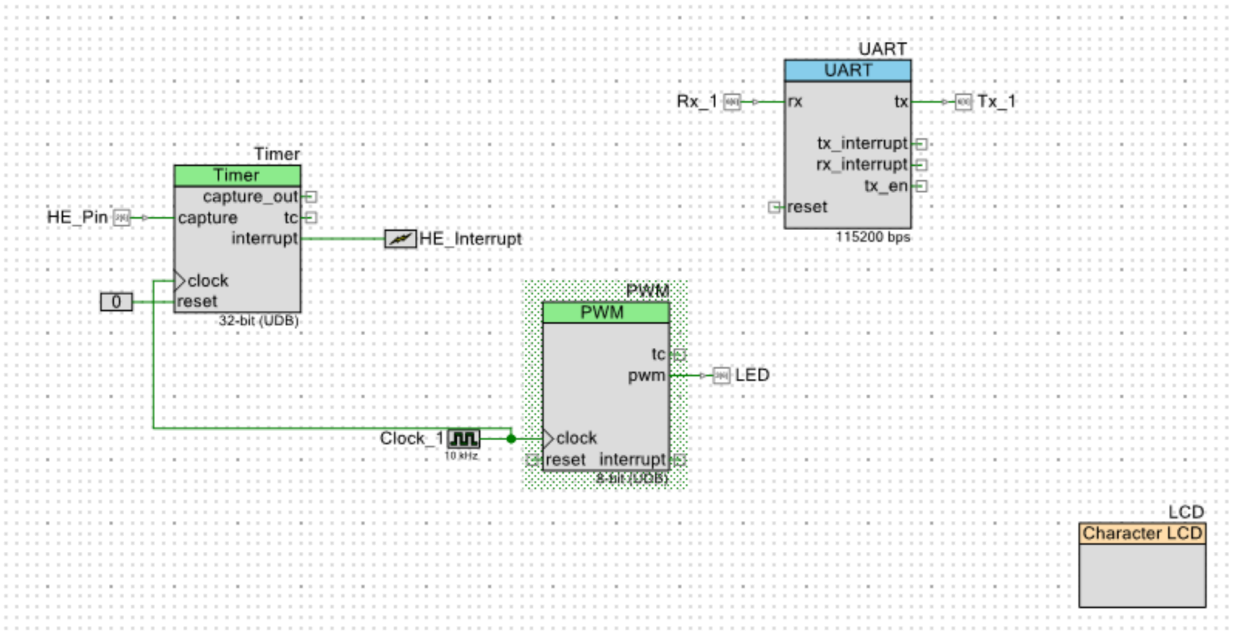
Other improvements we can make in the future are more organizational. In terms of wire colors, any wire that is connected to the motor should be either black or red. And for other parts (like the PSOC, voltage regulator, hall effect) should be used with other colors. This makes it easier to debug in the future if there are any hardware issues going on. Additionally, before soldering all the parts onto the PCB, we want to space the circuit components well so that when we solder we don’t short circuit and also so that there is enough space to include all components (we had a case where the heat sink and the kk connectors were tightly aligned). Additionally, we want to solder in iterations so that we do not have to solder everything at once or there is a risk that components will fall out and the process becomes complex and disorganized.

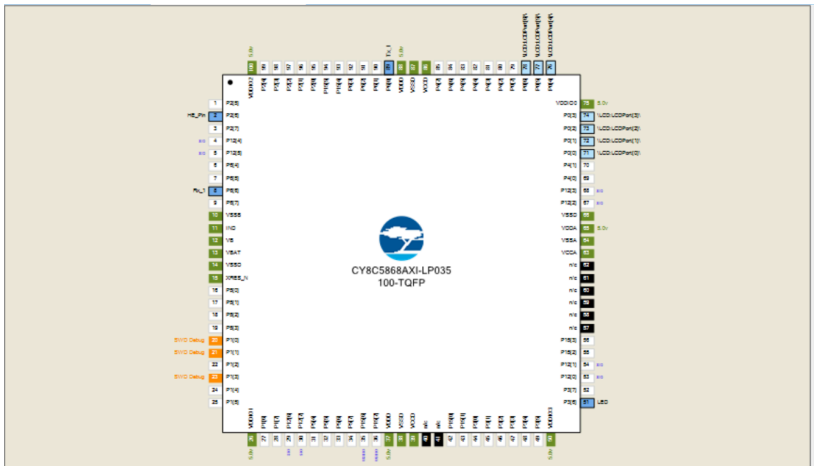
7. Code

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include <stdio.h>
#include <math.h>

#define TARGET_SPEED_FTPS   4.0f
/* in inches (later, convert to feet by dividing by 12) */
#define WHEEL_RADIUS        1.25f
#define MAGNET_COUNT        5
#define PI                  3.14159265f

// circumference in feet = (wheel radius in feet) * (2 * pi)
static const float WHEEL_CIRC_FT = (2*PI / 5) * (WHEEL_RADIUS /
12.0);
// static const float MAX_ERROR = 100;
// static const float MIN_ERROR = -100;

double error;
uint32 elapsed_time;
uint32 last_time;
uint32 curr_time;
double elapsed_cycles;
double speed;
double kp = 30;
double ki = 5;
double int_error;
double base = 30;
double pwm;

// Duty cycle:
char strbuf[16];
```

```
CY_ISR(inter)
{
    curr_time = Timer_ReadCapture();
    elapsed_cycles = last_time - curr_time;
    last_time = curr_time;
    speed = (WHEEL_CIRC_FT / elapsed_cycles) * 10000;

    error = TARGET_SPEED_FTPS - speed;
    int_error += error;

    pwm = base + (kp * error) + (ki * int_error);
    // if (pwm <= 0) {
    //    pwm = 20;
    //}

    //if (error > MAX_ERROR) {
    //    PWM_WriteCompare(200);
    //}
    //else if (error < MIN_ERROR) {
    //    PWM_WriteCompare(20);
    //}
    //else {
    PWM_WriteCompare((uint) pwm);
    //}

    sprintf(strbuf, "Speed: %3.0f", speed);
    LCD_Position(0,0);
    LCD_PrintString(strbuf);

    char buffer[64];

    sprintf(buffer, "%d,%d,%d,%d,%d,%d\r\n",
        (int)(speed*100), (int)(error*100), (int)(int_error*100),
(int)(pwm*100), (int)(elapsed_cycles), (int)(curr_time));
    UART_PutString(buffer);
    //UART_PutString("Hello from PSoC\r\n");

    Timer_ReadStatusRegister();


}


int main(void)
```

```
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g.
MyInst_Start()) */
    LCD_Start();
    PWM_Start();
    UART_Start();

    Timer_Start();

    HE_Interrupt_Start();
    HE_Interrupt_SetVector(inter);


    for(;;)
    {
        /* Place your application code here. */
        //UART_PutString("Hello from PSoC\r\n");
    }
}

/* [] END OF FILE */
```





Name	Port	Pin	Lock
\LCD:LCDPort[6:0]\	P0[6:0]	78…76,74…71	☐
HE_Pin	P2[6]	2	☑
LED	P3[6]	51	☑
Rx_1	P6[6]	8	☑
Tx_1	P6[0]	89	☑