Speed Control Report

ECE302 - Robotic & Autonomous Systems
Group 308
10/3/25
Callista Chong: cc0225@princeton.edu
Mayan Wasu: mw8270@princeton.edu

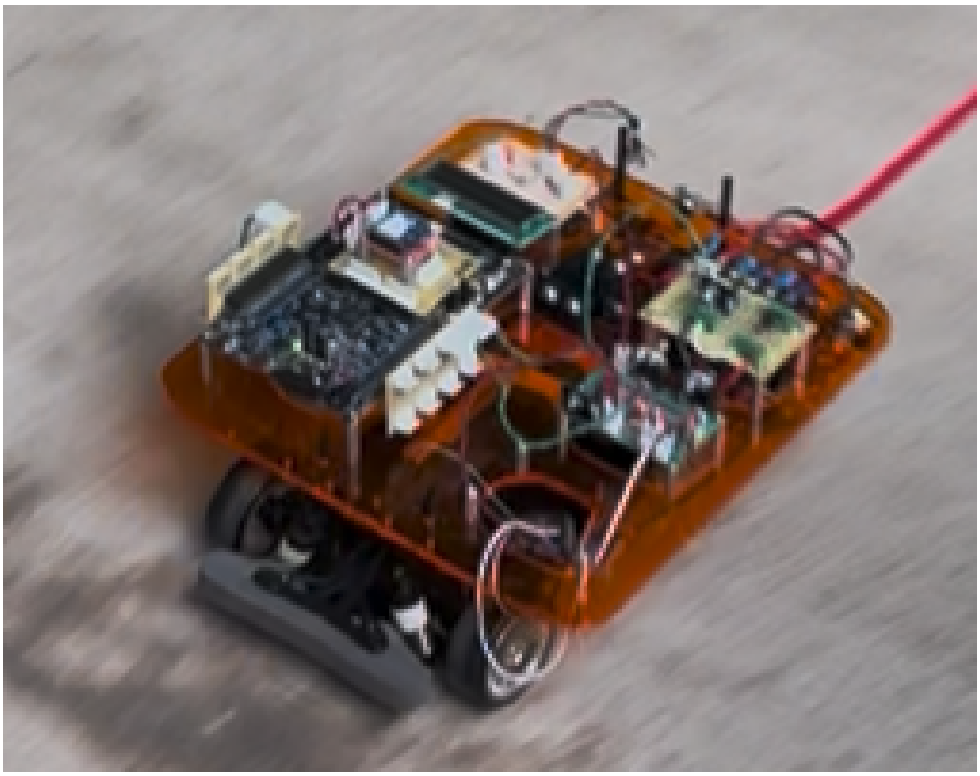

Figure 0: The Chongasu-Mobile

Table of Contents:

Statement of Objective

The primary objective of this project was to design, build, and program a car capable of navigating three distinct terrains (flat ground, an upward slope, and a downward slope) while maintaining a constant speed of 4 feet/second. The car was required to traverse 40ft of flat ground in approximately ten seconds with a tolerance of ± 0.2 seconds. For the slopes, the car was to complete the course within 8.1 to 9.9 seconds. These goals were achieved through hardware design and building, and careful PID control was implemented through the PSoC.

Overview

The car is comprised of five key subsystems: a power distribution system, a voltage regulator board, a power MOSFET board, a hall-effect board, and a video board (not used for this segment). In addition to these subsystems, the car is fitted with a DC motor, a servo, and a PSoC.

At a high level:

- The power distribution board delivers 7.2 V DC to the MOSFET/motor path and 9.6 V DC to the voltage regulator/PSoC path.
- The voltage regulator board generates regulated 5 V and 6 V rails for sensors and the servo.
- The power MOSFET board drives the motor using a PWM signal from the PSoC.
- The hall-effect board provides real-time speed feedback from wheel-mounted magnets for PID tuning

1. Power Distribution

Schematic:

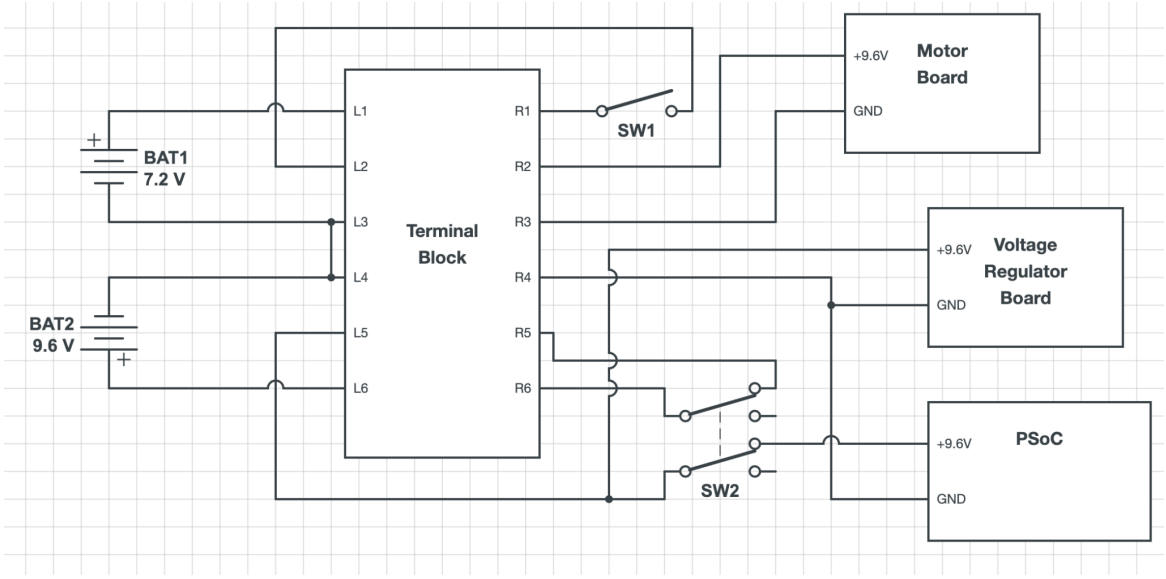

Figure 1: Schematic view of connections to the terminal block

Overview:

The power distribution board uses a terminal block to split a 9.6 V battery and a 7.2 V battery across subsystems. The 7.2 V is routed directly through the Power MOSFET Board to the motor. The 9.6 V battery powers the PSoC and the Voltage Regulator Board, where it is regulated to 5 V (for sensors) and 6 V (for the servo). The 9.6 V pack is rated 2200 mAh, chosen because the control electronics draw relatively low current. Contrastingly, the 7.2 V pack is rated 4800 mAh, chosen because the motor draws significant current. It is also important to note that having separate batteries for these two subsystems is beneficial because it isolates the noisy motor transients from the sensitive logic/sensor circuitry.

Now we can look into the switching systems:

- The 7.2 V circuit uses a single-pole single-throw (SPST) switch. This is just a conventional switch to turn the motor ON/OFF.
- The 9.6 V circuit uses a double-pole double-throw (DPDT) switch. This configuration prevents backfeeding between the PSoC's USB supply and the regulator board's supply during programming.

Data:

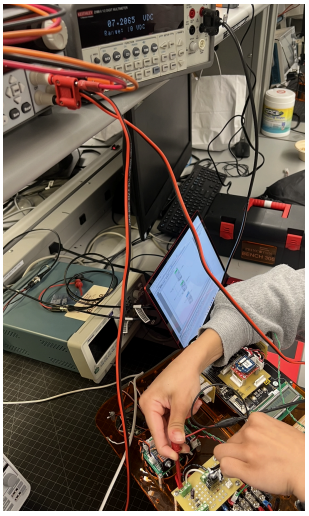

Figure 2(a): Motor Board

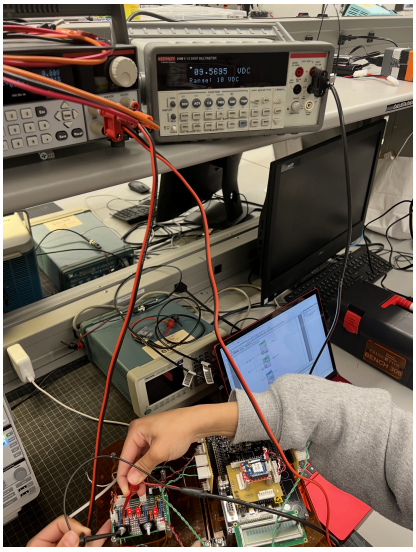

Figure 2(b): Voltage Regulator Board

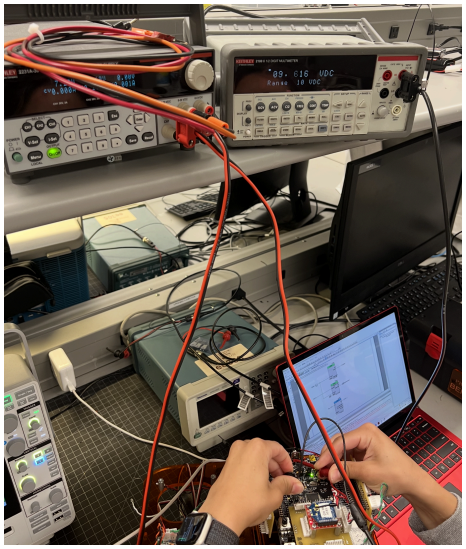

Figure 2(c): PsoC

The Power Distribution board effectively distributes battery power to subsystems. We can ensure this visually by probing across each subsystem's input (corresponding to a direct output of distribution system) with a Kiethley to observe that each system is getting the expected voltage.

2. Voltage Regulator Board

Schematic:

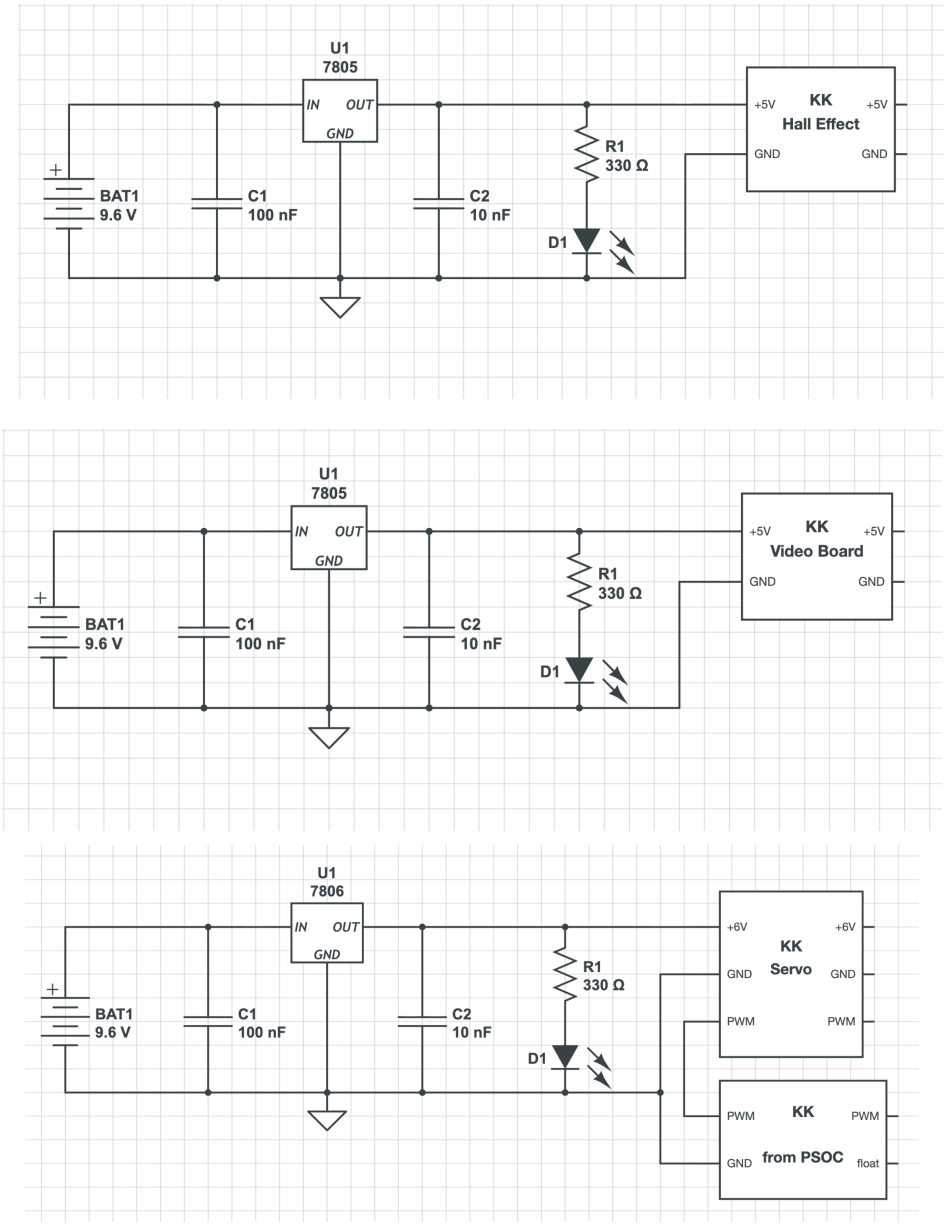

Figure 3: Schematic view of HE, Video, and Servo subsystems of Voltage Regulator Board

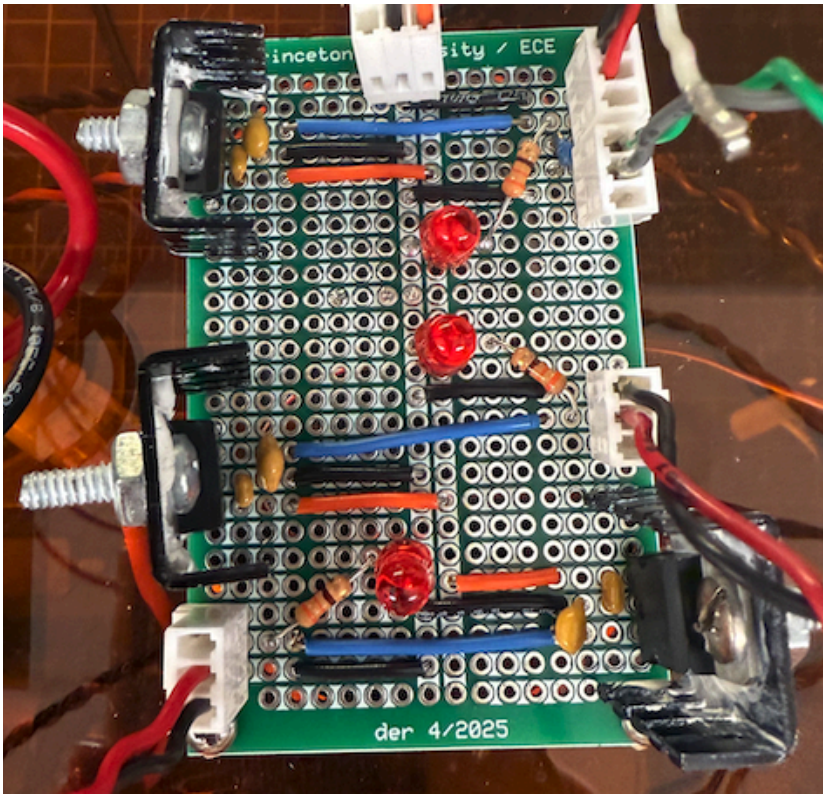

Figure 4: Top View of the Voltage Regulator Board

Overview:

The Voltage Regulator Board takes in +9.6 V DC and produces two +5 V rails and one +6 V rail, using LM7805 and LM7806 linear voltage regulators with small ceramic capacitors placed close to the regulator pins to keep them stable and to reduce high-frequency noise. Linear regulators (78xx series) maintain a fixed voltage output by converting the excess voltage into heat. For this reason, we use a thermal paste to bind the regulators to a heat sink to assist thermal dissipation and prevent overheating.

Each regulator stage also has:

- C1 (0.1 µF) and C2 (0.01 µF) capacitors to stabilize the regulator and suppress oscillations.
- Red LEDs in series with a limiting 330 Ω series resistor to ensure that the 9.6V battery has a high enough voltage to operate the regulators and feed the subsystems, and also function as convenient visual indicators of power.

Data:

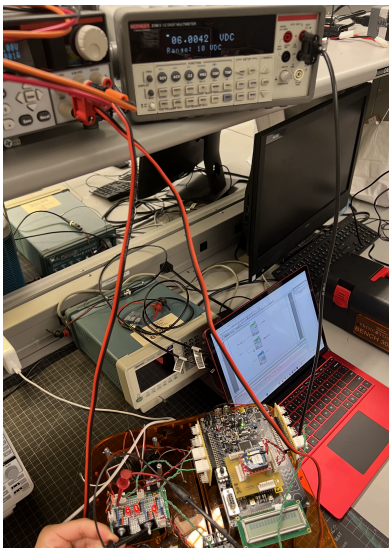

Figure 5(a): Servo

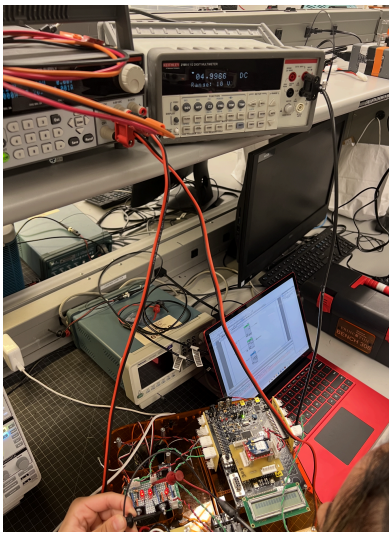

Figure 5(b): Video

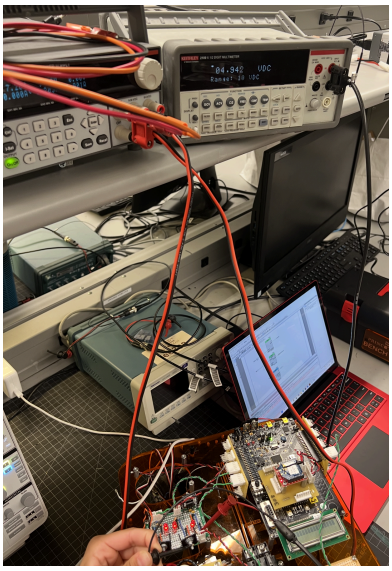

Figure 5(c): Hall-Effect

The Voltage Regulator board effectively regulated the provided 9.6V down to 5V and 6V for the appropriate subsystems. We can ensure this visually by probing across each subsystem's input path with a Kiethley to observe that each system is getting the appropriate voltage. Also, the LEDs built into the system ensure that we can see when a high enough voltage to work with the voltage regulators is flowing through each path.

3. Power MOSFET Board

Schematic:

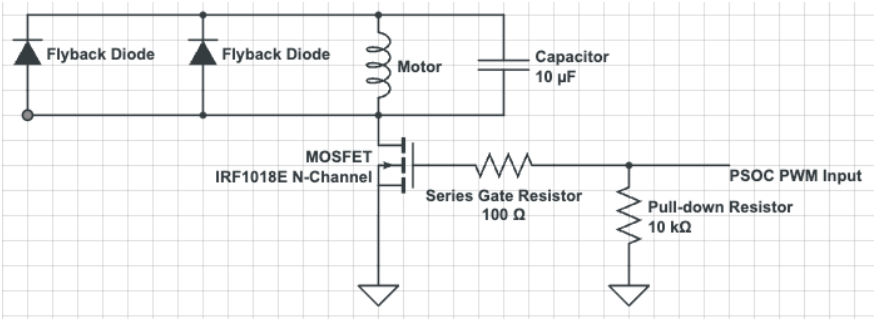

Figure 6: Circuit Diagram of the Motor Board

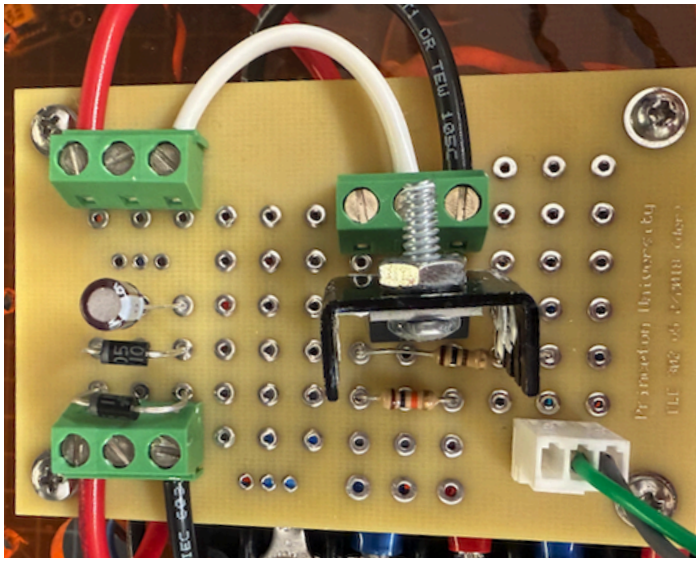

Figure 7: Top View of the Motor Board

Overview:

The MOSFET board controls the DC motor. A brushed DC motor is modeled as an inductor in series with a resistor, because its windings store energy magnetically.

The motor is placed on the drain side of the MOSFET, so the MOSFET acts as a low-side switch with the source tied to ground, the drain connecting to the motor's negative terminal, and 7.2V from the power distribution system on the motor's positive terminal. The MOSFET gate is stimulated by a PWM signal generated by the PID algorithm.

The board is comprised of the following elements:

- Two Schottky flyback diodes (fast switching, low forward voltage) placed across the motor terminals help to clamp the inductive kick. These are necessary to protect the MOSFET during fast switching events.
- 10 µF polarized electrolytic capacitor across the motor. This creates an LC tank to help dampen voltage spikes and supports the diode action.
- R1 = 10 kΩ pull-down resistor. This acts as a sink to ground and ensures the MOSFET gate is held low when not actively driven.
- R2 = 100 Ω series gate resistor. This limits inrush current into the MOSFET gate during switching and protects the MOSFET.
- IRF1018E N-channel MOSFET (Vth ~ 2.0–4.0 V, Rds(on) = 11 mΩ.) We selected a MOSFET that had a threshold voltage below the output high level of the PSoC PWM so that we would exit subthreshold easily when the device was being supplied. This particular device was also very resilient, with a high maximum input voltage and high current handling.

Data:

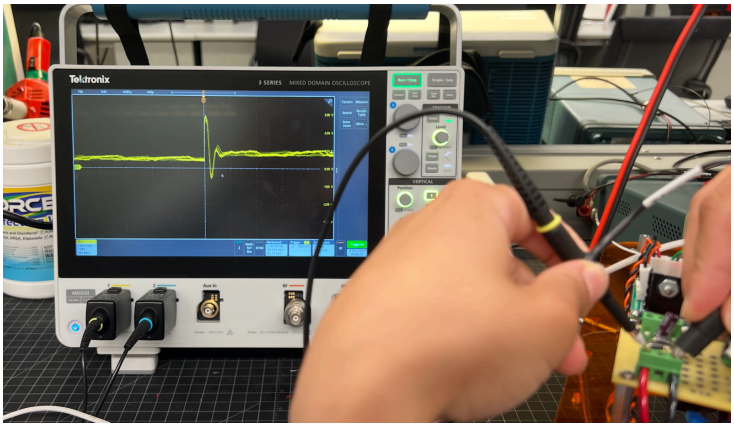

Figure 8: Voltage across Motor 1 and Motor 2

The Power MOSFET Board is effective in taking in a PWM to power the cars motor. We can ensure this visually by probing across the motor pins to see that a potential drop is being generated by the system. We can also assure that the board is working properly since the motor turns on!

4. Hall Effect Board & Sensors

Schematic:

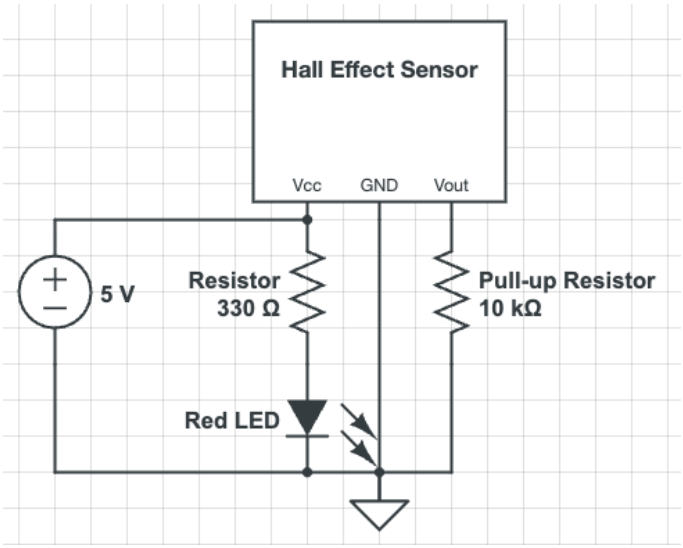

Figure 9: Circuit Diagram of HE Board

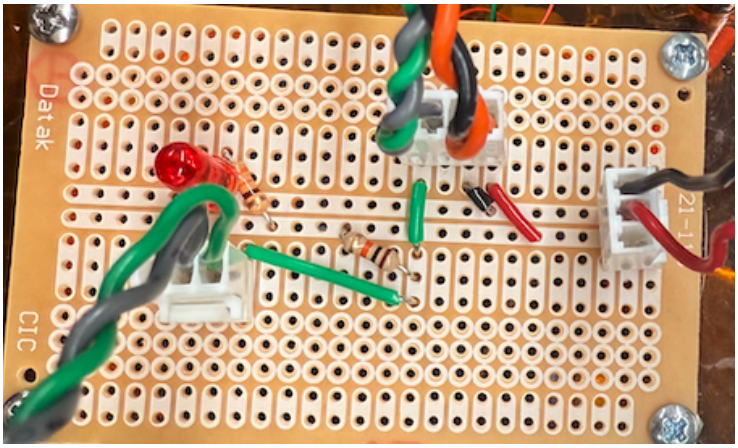

Figure 10: Top View of the HE Board

Overview:

The speed sensor uses an Allegro™ A110X digital Hall-effect sensor. We use this sensor in conjunction with 5 evenly placed magnets around the back of a wheel: an output was triggered as the wheel turned, and the sensor detected the magnet's electric field. Each pulse triggered an interrupt on a Timer module in the PSoC, which measures elapsed ticks. The sensor's output transistor saturates to GND when a magnetic pole is detected and floats when there is no detection. Since the sensor is open-collector, we add a 10kΩ pull-up resistor to tie the floating pin to 5V. On each rising edge (when we pass a magnet), the timer value is latched. With a known distance per pulse (one-fifth the circumference of the wheel) and a known timer frequency, we can calculate the car's speed.

Data:

The Hall Effect Board effectively provides a signal to the PSoC for speed control. We will go more in-depth about ensuring its function in conjunction with the speed control algorithm evaluation in the next section, where we trace the Hall Effect Board output for different speeds and compare that to the trace of the PWM signal sent out by the PSoC for each of these speeds.

XBee Data & Analysis

Diagram and Overview of Speed Control

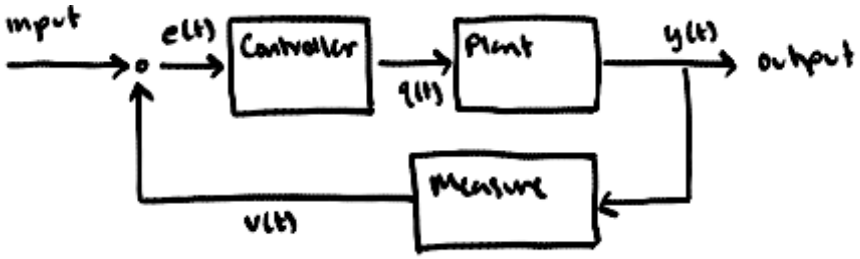

Figure 11: Conceptual Sketch of Proportional Speed Control Paradigm

For speed control, we used a discrete PID controller as in *Figure X*. The error is the reference 4 ft/s minus the measured speed. The proportional term Kp provides the main corrective push, the integral term Ki removes steady state error from hills and friction, and the derivative term Kd damps overshoot. We began with Kp only and increased the proportional gain until we reached an optimal value that minimized the time it took to settle at the target speed. This worked well for us on flats, but performed poorly on uphills and downhills. To correct this, we added our integral term and tuned the values of Ki. This integral term acts to reduce accumulated error as opposed to the instantaneous error that the proportional term addresses. The Ki term allowed us to return to the reference speed faster on our uphill and downhill slopes. Integration is effectively suspended when the PWM saturates or when the car is nearly stopped. A small derivative term, Kd, could have been added to shorten settling, but through careful tuning of Ki and Kp, our car was performing up to standards on all three terrains, so it was not necessary to introduce. This produced fast convergence on the flat, strong disturbance rejection uphill, and stable behavior downhill without runaway.

System Validation

To ensure that the sensitive devices and algorithms involved in this system are all interacting appropriately, we will capture scope traces of the output from the Hall-Effect Board and corresponding PSoC output PWM using the oscilloscope for three different reference speeds.

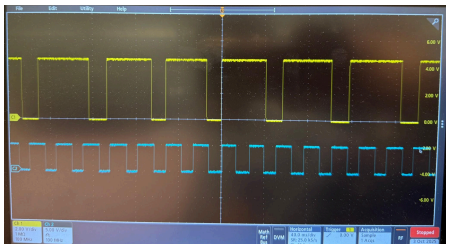

Figure 12(a): 3 feet/s

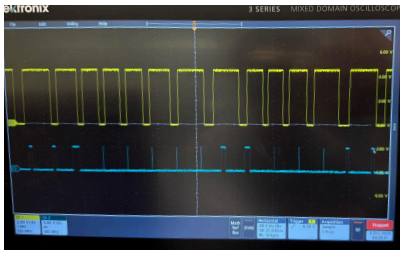

Figure 12(b): 6 feet/s

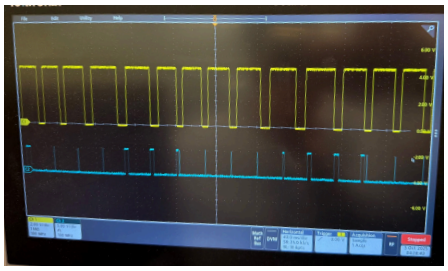

Figure 12(c) 9 feet/s

From these images, we can see how increasing the speed of the wheels turning triggers the HE output (yellow) at a higher frequency, which corresponds to the output PWM (blue) having a higher frequency of pulses. The Hall Effect system produces a very uniform trace, meaning the sensor and magnets were placed diligently, but the PWM gets a little bit modulated as the frequency increases. We do not operate above 4ft/s, though, so this modulation does not affect the operation of our vehicle within its designated threshold of performance.

Variable Tuning

Once the connectivity and reliability of the system was confirmed, it was time to move on to tuning the PID control variables (PWM base, kp, ki, and kd). We moved through these iteratively, trying to achieve peak performance by only adjusting the PWM base before moving on to kp and ki. It was determined that the derivative term was unnecessary to add, since we had achieved performance requirements with just kp and ki.

PWM Base:

By adjusting the base PWM, our goal was to simply have the car reach 4 ft/s at a reasonable speed-up time. As evident in the provided figures, as the PWM base was increased, the steady state speed increased. Excluding noise that cropped up at lower speeds, there exists an undesirable, slow buildup to saturation. This will be mitigated by later tuning variables. For this stage, we just wanted the car to reach a consistent and dependable 4ft/s on flat ground.

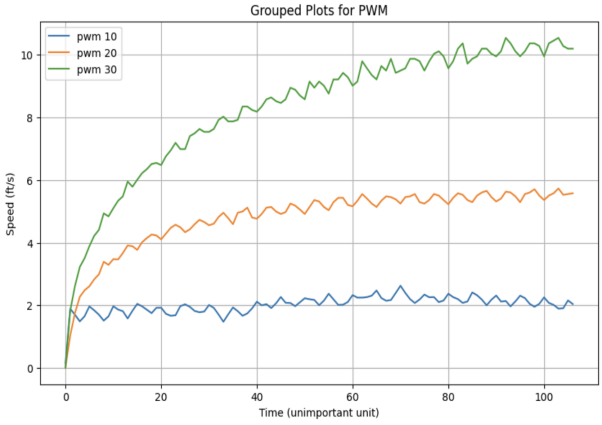

Figure 13: Speed Readings at 3 PWM Values

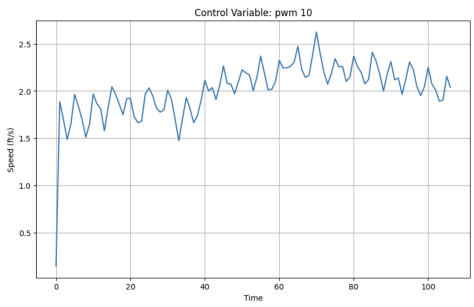

Figure 14(a): Speed Plot at PWM = 10

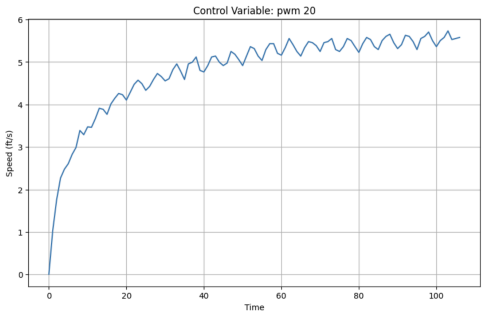

Figure 14(b): Speed Plot at PWM = 20

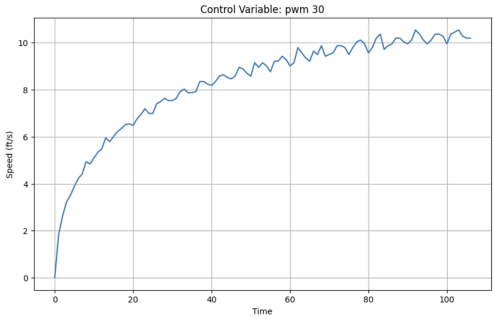

Figure 14(c): Speed Plot at PWM = 30

It is important to note that this data was taken from the car while the wheels were floating on the test bench, so the reflected speeds neglect friction. During ground testing, it was found that a pwm base of 24 performed optimally.

Kp:

Next was the proportional term, Kp. This tuning stage had the strongest immediate effect on the cars motion: at lower gain, the response was slow but stable, with minimal oscillation and a steady-state variance of only 0.029. However, the steady-state speed fell slightly (average 4.22 ft/s). Increasing Kp improves rise time but also introduces overshoot and oscillatory behavior, pushing steady-state variance an order of magnitude higher (0.325). At the highest tested value (Kp = 100), the system reached the target more quickly but became increasingly unstable, with very large fluctuations (variance >1.0).

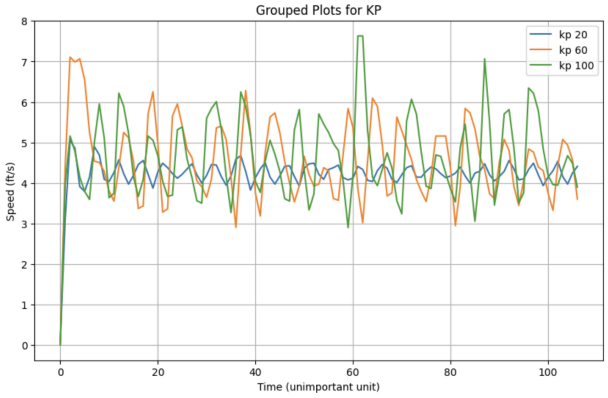

Figure 15: Speed Readings at 3 Kp Values

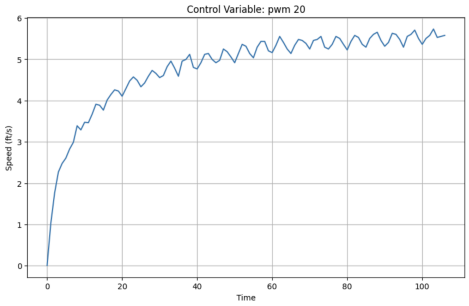

Figure 16(a): Speed Readings at Kp=0

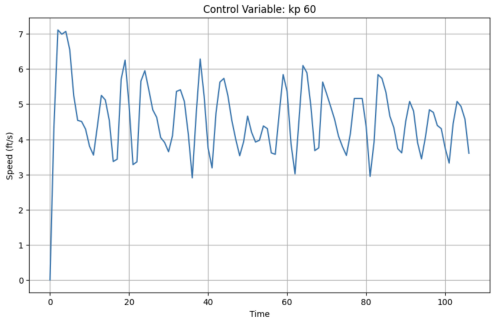

Figure 16(b): Speed Readings at Kp=60

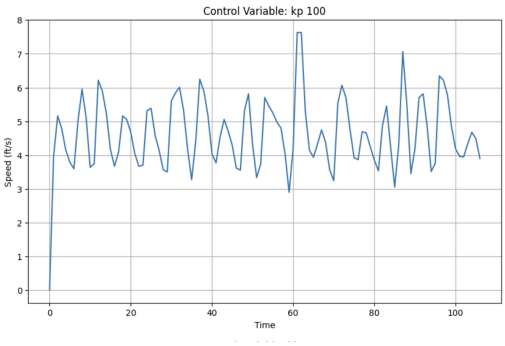

Figure 16(c): Speed Readings at Kp=100

From these figures, we can visually confirm that while larger proportional gains accelerate error correction, they also amplify noise and create oscillatory behavior. Thus, Kp must be chosen carefully to balance responsiveness with stability. For our car, the optimal value tested on flat ground and slopes was 60.

Ki:

The integral term, Ki, governs how accumulated error influences the system, gradually correcting for persistent offset. It is difficult to diagnose how Ki interacts with the system in the testing environment that this data was collected in (floating wheels, no slope), but it is still somewhat evident: with very small values, the controller tracked close to the target with moderate oscillations. Increasing Ki to 0.004 or 0.01 introduced stronger corrections for steady-state error, but the system became increasingly noisy. Interestingly, the average speeds for all Ki values remained close to the target (4.41–4.48 ft/s), suggesting that integral action effectively eliminated bias but at the cost of stability.

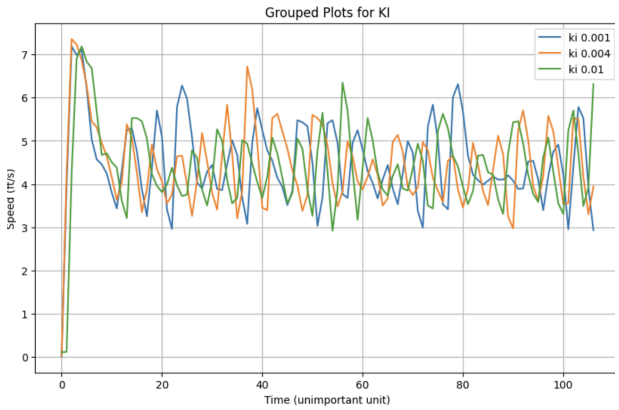

Figure 17: Speed Readings at 3 Ki Values

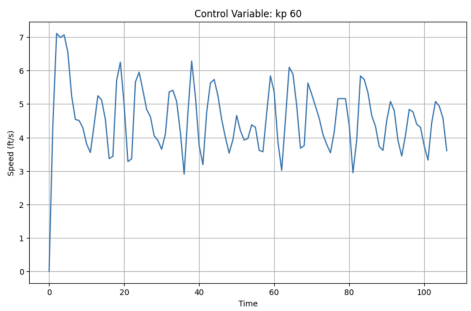

Figure 18(a):Speed Readings at Ki=0

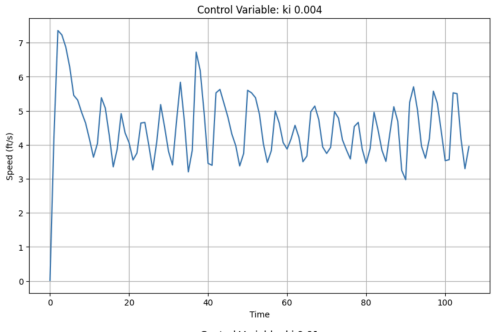

Figure 18(b):Speed Readings at Ki=0.004

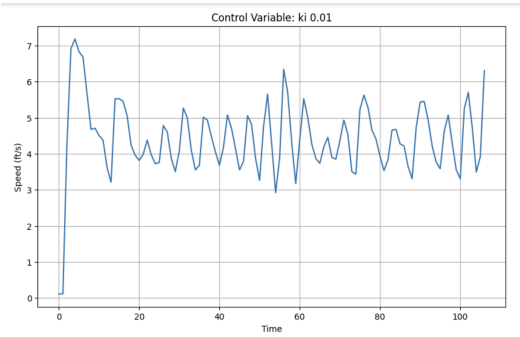

Figure 18(c):Speed Readings at Ki=0.01

Too little Ki leaves residual error, while too much amplifies oscillations and destabilizes the response. On a slope, when the car is consistently going a little too fast or too slow, if Ki is too big we can get massive overshoots and error accumulations. A small integral term, paired with an appropriately tuned proportional gain, provided the best compromise. For our car, this value was 0.004.

Final Variable Validation:

After careful tuning, we need to test the performance of the completed system. To achieve the given performance requirements, our car needs to traverse the flat, uphill and downhill tracks while maintaining an approximate speed of 4 ft/s.

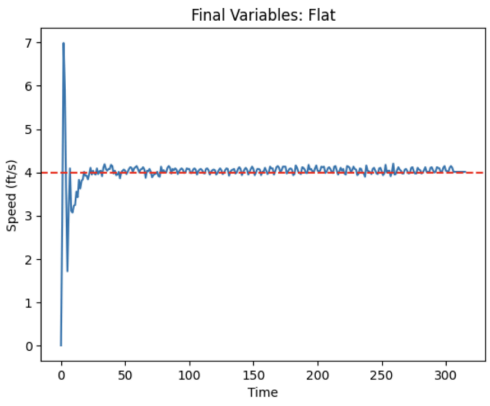

Figure 19(a): Final Speed [Flat] Avg: 4.02 ft/s

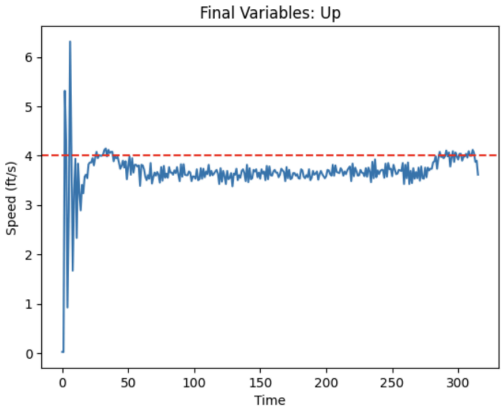

Figure 19(b) Final Speed [Up] Avg: 3.68 ft/s

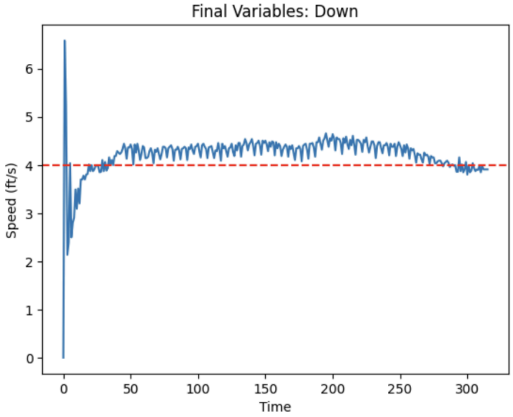

Figure 19(c) Final Speed [Down] Avg: 4.21 ft/s

Visual inspection as well as numerical calculation confirms that the car maintains an approximate speed of 4 ft/s on all terrains, with higher error on the slopes. We can also see in each graph how the tuned parameters affect the speed: each graph has small but persistent oscillations (while maintaining acceptable stability) and corrects really well as the run goes on. At the end of the operating period for both the uphill and downhill tracks, the speed converges on 4 ft/s. This shows that out tuned variables are functioning as expected!

Challenges & Discussion

Some alterations we made to our designs before our first iterations were including two Schottky diodes in parallel across the motor. Using two diodes in parallel across the motor allowed them to share current and clamp both positive and negative transients more effectively than a single device, protecting the MOSFET's body diode from back-EMF that was caused due to the motor acting as an inductor.

We also used two capacitors for our voltage regulator board: two ceramic capacitors placed close to the regulator pins. The larger 0.1 µF capacitor stabilizes the regulator against mid‑frequency oscillations while the smaller 0.01 µF capacitor filters very high‑frequency noise that could otherwise couple into the logic rails. This choice was mirrored across the two +5 V outputs and the +6 V servo supply. Without the dual‑capacitors, the regulators exhibited oscillations when we probed them.

One of the first challenges that came after its first iteration came from our power MOSFET stage. During high duty cycle bench tests, the MOSFET burned out which we were unaware of until we completed the other components and then tried testing the entire car. Diagnosing the burnt‑out MOSFET was a difficulty. By probing the drain and source voltages with an oscilloscope, we observed that voltage spikes were exceeding the device's safe operating range. After replacing the MOSFET, the system operated reliably. Our takeaway was that using oscilloscopes and probing iteratively over components in isolation is a really helpful and effective method to diagnose problems.

Finally, tuning the controller was itself a challenge. We experimented with variable values of Kp, Ki, and Kd on both the flat, the up-slope, and the down-slope. The process highlighted the tradeoff between responsiveness and stability: higher Kp gave faster rise but risked oscillation, Ki eliminated slope‑dependent bias, but risked over-correction over time. Initially, after our first round of tuning, the car performed well on both the flat and the up slope from a timing perspective. What we struggled with was our down-slope - the car would go too fast down the slope, and become extremely slow and sluggish as soon as it hit the flat after the down slope. What we realized was happening was we had settled on a value of Ki that was too high - errors would accumulate at a much faster rate than we could correct, but it didn't solve the mystery of why the car had a snail's pace as soon as

we hit flat ground. What we realized was that we had set the minimum PWM to be too high - this meant that the error correction on the downslope could not force the car to slow down by setting any PWM below 10. Once we reduced our base PWM and decreased the magnitude of Ki, the performance improved dramatically. Some further fine-tuning then led us to our final speed control algorithm which successfully traverses the flat, up-slope, and down-slope within the prescribed time ranges.

Further Improvements

Our PID control, although sufficient, could be further improved. We can explore adding the derivative term, which acts as the dampening component, anticipating rate of change errors to counteract overshoot and smoothing out oscillations. It is important to be weary of including a Kd term that is too high, since the system will become overly damped/sluggish and amplify noise.

There are also some mechanical and electrical components that can be optimized. We observed a persistent rightward drift in motion, which indicated structural issues. In addition, a few KK connectors (PWM from PSoC, HE to PSoC) are unreliable and had to be reconnected a few times during testing. It is worthwhile to remake those connections to be more stable. Since PID relies on accurate feedback, even minor connection inconsistencies can destabilize the system. Also, the DPDT switch needs to be flipped to match the conventions of the SPST.

Code

Commented C Code:

```
#include "project.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>



#define CLOCK_HZ 10000UL // SpeedTimer clock (10 KHz -> 100 µs ticks)

#define TIMER_MAX 4294967295 // 32-bit timer (2^32 - 1)

#define MAGNET_SPACING_FEET 0.12438 // Distance per HE pulse in feet (2pi *
1,3/16inches)/5

#define TARGET_SPEED_FPS 4.0   // Desired speed in feet/sec
```

```
#define PWM_MIN 1

#define PWM_MAX 230


// variables we need to track times from HE sensor

volatile uint32 oldSeen = TIMER_MAX;

volatile uint32 newSeen;

double elapsedTicks;


// variables needed for speed and error

volatile double speed;

volatile double error = 0;

volatile double prevError = 0;

volatile double integralError = 0;

volatile double derivativeError = 0;


// variables for PID - the proportional constant and the integral term

double kp = 60;

double ki = .004;


// Initial PWM value (base speed) to MOSFET

double pwm;

double pwmBase = 24;


// Used to help print to debug and collected data

char print_speed[32];

char print_error[32];


char dbg[64];


// This interrupt will trigger every time magnet passes HE sensor

CY_ISR(inter){
```

```
        // captures the time from the timer and calculates the elapsed time from the
previous capture

        newSeen = Timer_ReadCapture();



        // Calculating how many ticks elapsed between magnet sighting

        elapsedTicks = (double) oldSeen - (double) newSeen;



        oldSeen = newSeen;



        // speed is calculated based on the elapsed time and the distance between
magnets on the wheel

        speed = 1243.8 / elapsedTicks;

        error = 4.0 - speed;



        // print for XBee Data Collection

        sprintf(print_speed, "%f", speed);

        UART_PutString(print_speed);

        UART_PutString("\r\n");



        // prevents error being added to the accumulated error if car stationary for a
long time

        if(elapsedTicks <=5000){

            integralError += error;

        }



        prevError = error;



        // new pwm is calculated based on the error

        pwm = pwmBase + kp * error + ki * integralError;



        // bounds the pwm values to the upper and lower limits
```

```
if (pwm>PWM_MAX)

pwm = PWM_MAX;

if (pwm<PWM_MIN)

pwm = PWM_MIN;

// the new pwm signal is sent to the motor board

PWM_WriteCompare((uint8)pwm);

// clears interrupt

Timer_ReadStatusRegister();

}


int main(void)

{

// Initialize components + Enable interrupts

CyGlobalIntEnable;

Timer_Start();

PWM_Start();

HallInterrupt_Start();

UART_Start();

HallInterrupt_SetVector(inter);

// Main control loop

for(;;)

{


}

}

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