Abstract

The goal of this project was to build a nearly fully analog miniature segway. The segway has principal components: motor drivers, a gyroscope, a servo signal decoder, and feedback circuitry. In order to control the motors, I am using two H–bridge motor drivers with current feedback. These H–bridge motor drivers are controlled by a state space controller that measures the angular acceleration from the gyroscope and feedback from the motors to keep the cart upright and moving at the desired velocity. Unfortunately, a reasonably priced analog gyroscope could not be found, but a digital gyroscope connected to an Arduino Nano with an I^2C DAC provides a pretty good approximation to a true analog gyroscope. The segway itself is made of laser cut acrylic, and it is powered by a 9.9 V lithium iron phosphate battery. I was able to successfully stabilize the cart and control it with a stock radio controller, driving it forwards and backwards and turning the cart left and right. Overall, this project successfully implemented a complex feedback system and state estimator entirely in analog electronics and used this feedback system to drive and stabilize a real system.

1 System overview

An extremely high level overview of this project gives four key components–the feedback circuit, the gyroscope, the servo signal decoder, and motor drivers. The most critical part of the segway is the feedback controller. The feedback controller interfaces with the other three components, taking inputs from all three and outputting a target current to the motor controller. All signals in the circuit range from 0 to 5 volts, with a virtual ground at 2.5 V. This means that all voltages that correspond to numerical values should be interpreted

relative to 2.5 V.

The feedback controller receives three voltages from the gyroscope unit corresponding to rotation rates around the y and z axes and acceleration along the x axis. It also receives a voltage corresponding to the motor controller PWM duty cycle from the motor controllers, and voltages corresponding to the target velocity and z rotation rate from the servo signal decoder. These voltages are fed into estimators for angle and velocity of the cart, and then those estimators, gyroscope voltages, and target voltages are mixed to give commands to the motor controllers. The motor controllers are current controlled, so an input voltage corresponds to the shunt resistance of the motor controller times the target current.

2 Feedback controller design

2.1 The inverted pendulum controller

A mini segway is an inverted pendulum, and it can be physically modeled as such. Let θ be the angle of the cart measured from vertical, x be the cart position, and F be the horizontal force applied by the motors.

We can describe a linearized open–loop model of the cart around one axis as

$$\dot{\mathbf{q}} = \mathbf{A}\mathbf{q} + \mathbf{B}F,$$

where $\mathbf{q} = \begin{bmatrix}\theta, & \dot{\theta}, & \dot{x}\end{bmatrix}^T$, and A and B characterize the system. A more detailed derivation of values for A and B can be found in Appendix A.

In order to stabilize this system, we can create a feedback system with $F = \mathbf{K}\mathbf{q}$, where $\mathbf{K}$ is the gain vector. We tune the value of $\mathbf{K}$ so that the system is stable. In short, we send

a weighted sum of velocity, angular velocity, and angle to the motor controller in order to stabilize the cart.

One interesting consequence of this controller is that it essentially uses positive feedback for velocity. In order to move the cart forwards, a backwards force must first be applied to the bottom of the cart, causing it to tip forwards. The angle feedback then kicks in, allowing the cart to accelerate forwards without tipping over.

2.2 Estimators

Our controller described in the previous section relies on θ and $\dot{x}$, both of which are not accurately directly measurable. With enough knowledge of the parameters of our system, we can use Kalman filters to accurately predict both θ and $\dot{x}$.

We will define g_x, g_y, g_z to be the measured angular velocities of their respective axes, and a_x, a_y, a_z to be the measured accelerations. The cart's coordinate system is defined such g_y is equal to $\dot{\theta}$ and a_x is roughly in the same direction as $\ddot{x}$.

Our estimator for θ is based on the fact that,

$$a_x = -g \sin\theta + \ddot{x} \cos\theta.$$

We know that $\dot{x}$ is bounded, so we can make the simplifying assumption that $\ddot{x} = 0$ when calculating $\hat{\theta}$. Using the small angle approximation $\sin(\theta) \approx \theta$, we can create the estimator

$$\hat{\theta} = \int g_y - K_{\hat{\theta}} \left(\hat{\theta} + \frac{a_x}{g} \right) dt,$$

where g is $9.8\,\mathrm{m/s^2}$ and $K_{\hat{\theta}}$ is the Kalman gain. Experimentally, we found that a Kalman gain of about 0.3 works best.

The estimator for $\dot{x}$ is more complicated. Using the above formula and approximating $\cos\theta \approx 1$, we have $\ddot{x} \approx g\theta + a_x$. We can further improve the estimator by noting that the IMU is mounted on top of the segway, giving $\ddot{x} \approx g\theta + a_x - \ddot{\theta}L$, where L is the distance from the center of rotation to the IMU. For this segway, $L = 30\,\text{cm}$.

Using the above equations, we choose the estimator

$$\hat{v} = -Lg_y + \int a_x + g\hat{\theta} - K_{\hat{v}}\left(\hat{v} - K_e V_{emf}\right) dt$$

with K_{hatv} as the Kalman gain and K_e is a constant that relates motor back emf to velocity. In practice, we can omit the $-Lg_y$ term from the estimator and decrease the gain for g_y in the controller, and we found that $K_{\hat{v}} \approx 0.3$ works very well.

2.3 Practical implementation

Since true analog gyroscopes are relatively expensive, an Arduino Nano was used to read an MPU6050 gyro/accelerometer unit and output the measured values to an MCP4728 four–channel I2C DAC. The outputs are configured such that the measurements are centered around 2.5 V with 100 mV/(rad/s) for the gyroscopes and 100 mV/(m/s^2) for the accelerometers.

In order to tune the system and find state–space gains that worked well, I simulated the control system on the Arduino and spent some time tuning the gains for a fast step response with low overshoot. I also tried several different estimators before settling on the estimators for $\hat{v}$ and $\hat{\theta}$ used above.

An op–amp integrator with a resistor in parallel to the feedback capacitor can be used to implement the estimators for $\hat{\theta}$ and $\hat{v}$. Increasing the value of the feedback resistor decreases

the Kalman gain. For the velocity estimator $\hat{v}$, the PWM reference voltage in the motor driver was used as well as the angle estimator $\hat{\theta}$ and the accelerometer output. For the $\hat{\theta}$ estimator, an output resistance of 300 kΩ and a 10 µF bipolar electrolytic capacitor give a time constant of about 3 s, and 10 kΩ and 300 kΩ resistors are used for the inverted gyroscope signal and accelerometer signal, respectively. The $\hat{v}$ estimator also uses a 300 kΩ feedback resistor and 10 µF capacitor for a time constant of about 3 s.

The state–space controller is used as an inverting adder, and the gains for θ, $\dot{\theta}$, and v can be changed by changing the input resistors to the adder. One interesting consideration to be made was that the state–space controller actually has to output two motor signals–one for the left motor and one for the right motor.

In order to make the cart turn, we need to drive the motors at different velocities. I made a simple differential mixer by creating a signal with voltage $-v_{comm} - \frac{1}{2}v_{diff}$ with an inverting adder. A second non–inverting adder can be used to add v_{diff} to give $-v_{comm} + \frac{1}{2}v_{diff}$ on the other output. A simple proportional controller or PI controller can be used to control yaw and generate v_{diff}.

3 H–bridge motor drivers

In order to control the segway, my controller outputs voltage that corresponds to a torque on the wheels, and the best way to control torque is with a current–controlled motor driver. Without feedback, an H–bridge brushed motor controller will effectively control the voltage to the motor. The motors used are pretty low torque, so increasing the voltage of the motor will not immediately increase the speed, giving a sluggish response time. A current–

controlled H–bridge will actively change the PWM duty cycle to regulate the current through the motor. This configuration has the advantage that regardless of motor properties, the torque can be controlled very rapidly by varying the current through the motor.

To actually provide current to the motor, each H–bridge uses four STP36NF06L N-channel MOSFETs. These are relatively inexpensive power MOSFETS, but they are rated for 60 V V_{DS} and 30 A I_D, with a switching time of around 15 ns. An IR2101 half–bridge driver is used for each pair of MOSFETs. This gate driver is designed for high-side switching with a bootstrap capacitor, and since the switching frequency is around 50 kHz with a 1 nF gate capacitance, a 470 nF capacitor is a good choice.

The signals to the gate drivers are implemented by comparing a triangle wave to a reference voltage, and the resulting PWM signal is inverted with 74LS04 inverting buffers. The half bridges are driven completely out of phase, and the high and low sides of each half bridge are also driven out of phase. In this circuit, a 50 kHz triangle wave is generated with a 555 timer, and the reference voltage is generated by the feedback circuit.

Current feedback is implemented with a 0.33 Ω shunt resistor in series with the motor. The voltages at either end of the shunt are low–pass filtered and then sent into a differential amplifier. The output of the differential amplifier is fed into an integrator that controls the reference voltage to the comparator, setting the duty cycle. A 10 kΩ resistor and 0.1 µF capacitor were chosen for the integrator so that the motor current will approach the desired current in a few milliseconds.

4 Radio control

Building an entire radio transmitter/receiver system from scratch would be an extremely challenging project. To add radio control to my segway, I used a stock hobby radio transmitter and receiver. However, the output of the receiver produces a servo signal with an *absolute* PWM encoding—the pulse width varies from 1000 µs to 2000 µs, and the spacing between pulses does not matter. Decoding a PWM signal where the signal is encoded with duty cycle is substantially easier, because the signal can be recovered by low–pass filtering the received signal. In this case, I had to build an analog circuit that would convert the absolute duration of each pulse into an analog voltage and then store that voltage until the next pulse came in. I used an integrator to convert pulse width into a voltage and then used a MOSFET, capacitor, and op–amp as an analog sample–and–hold cell.

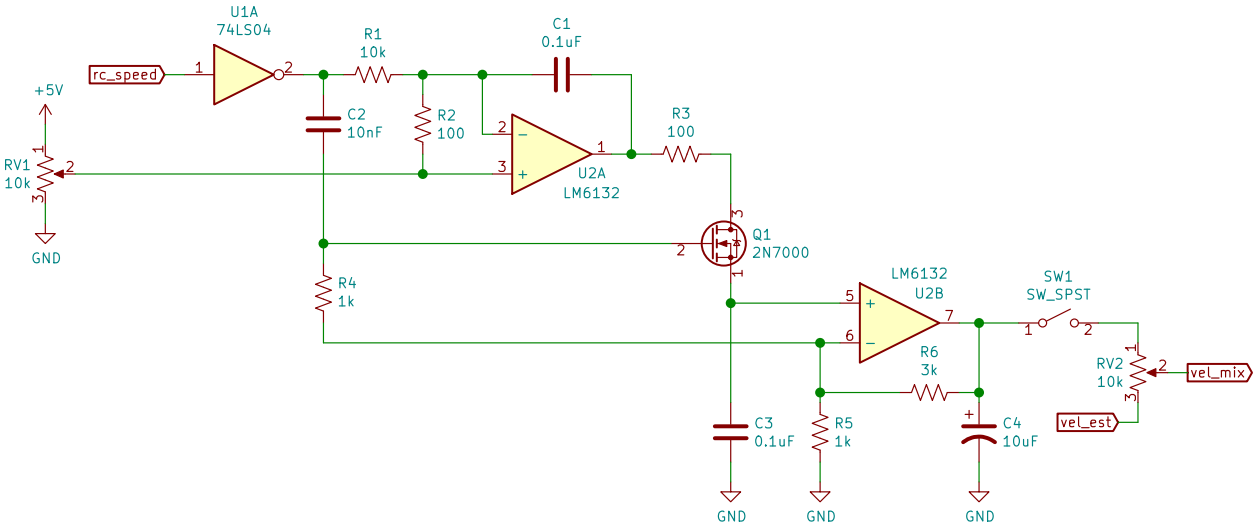


Figure 1: One channel of the receiver. U2A is an integrator, and Q1, C3, and U2B form a sample and hold cell.

Converting a pulse into an analog signal is achievable with an integrator—the integrator

voltage is linearly related to the pulse length. Since a simple op–amp integrator is inverting, I used a 74LS04 inverter to flip the input signal. Next, the inverted input is used to drive the gate of an N–channel MOSFET through a high–pass filter. Since the high pass filter is referenced to the gate of the FET, the FET is normally off. On the rising edge of the input (or falling edge of the inverted input), the FET gate voltage is driven below zero, keeping the FET off and keeping the sample–and–hold capacitor isolated from the integrator. On the falling edge of the input (or rising edge of the inverted signal), V_{GS} rises above the threshold voltage and the FET turns on, charging the buffer capacitor to the integrator voltage. After about 10 µs, the FET turns off, and the sample–and–hold op–amp holds a stable output voltage proportional to the capacitor voltage.

The positive input of the integrator op–amp is connected to a potentiometer so that the decoder will output 2.5 V when given a pulse width of 1500 µs.

One problem with this circuit is that the 2N7000 MOSFET has a flyback diode. This diode seems to have a forwards voltage drop of around 1.0 V, so a gain of 4 was added to the buffer so that the storage capacitor will not discharge through the flyback diode when the integrator resets itself to zero.

5 Additional hardware

Pure analog gyroscopes are unreasonably expensive, so I used an MPU6050 digital 3–axis gyroscope and 3–axis accelerometer chip connected to an Arduino Nano as an IMU sensor. In order to feed a voltage into my circuit, I used an MCP4728 4–channel I2C DAC. A reference voltage, the gyroscope y and z axis measurements, and the accelerometer x axis

measurement are outputted from the MCP4728.

The frame for the segway was laser–cut out of 1/4" acrylic sheet, and I designed it to have multiple breadboard shaped shelves. The motor controllers sit on the bottom shelves, and the control circuits and batteries are on the top two shelves. Pololu brushed gear motors were used for driving because they are easy to control and supply adequate torque to drive the wheels directly. A 9.9 V lithium iron phosphate battery was used to provide power to the whole circuit.

6 Challenges

6.1 Current controlled H bridge

One of the most difficult parts of building the H–bridge motor drivers was noise mitigation, especially with the supply rails. When the MOSFETs switch, there are large fluctuations in the rail voltages on the order of 100 ns in duration. The LM311 comparator used for generating the PWM signal fed to the gate drivers was relatively immune to this noise, especially since I used positive feedback with a series resistor and capacitor to prevent ringing from causing double–switching. Unfortunately, the supply noise was able to pass through the LM7805 voltage regulator and destabilize the control circuitry and 555 timer. I eventually settled on putting a 10 Ω resistor between the positive supply rail and the LM7805 input and putting a 10 µF capacitor between the LM7805 input and ground. This effectively formed crude RC low–pass filter that prevented most of the power–supply noise from passing into the control circuit. Some noise still found its way into the control circuit rails through capacitave

coupling, but it was not bad enough to disrupt operation.

Since the motor controller has to deal with relatively high currents, and since critical components slipping loose could cause potentially dangerous short circuits, the final version of the H bridge was soldered in place. This meant that mistakes were relatively difficult to fix, and the design had to be finalized by the time the circuit was soldered. One extremely important takeaway from hand–soldering an entire motor controller is that circuits should probably be tested with a current–limited power supply before plugging them straight into a LiPo battery rated for 60 A continuous output. Nothing caught fire, but one of the MOSFETs burned out.

6.2 State space controller

The process of designing the state space controller and estimators for velocity and angle was a huge challenge. When modelling the system, I was able to create a fairly accurate linearized model of the system. However, I was not able to extract very useful gains for the real physical system due to the difficulty in measuring properties of the real segway, such as moment of inertia and exact electrical characteristics of the motor. The model was very useful for providing rough guidelines for choosing gains, even if it did not make exact predictions.

When making the model, I neglected to model the observers for angle and velocity. It turns out that the construction and reliability of these observers is critical to the stability of the cart. Building a good angle observer is relatively simple and can be done by Kalman filtering the measured rotation rate and acceleration. The only major variable to control

with the angle estimator is the Kalman gain, which effectively controls the speed at which the angle estimate converges to the accelerometer average measurement. This estimator can be slightly improved by low–pass filtering the accelerometer before feeding it into the estimator, but the improvement does not have a very big effect on performance.

Building the velocity estimator is substantially harder. The inertial acceleration can be approximated by subtracting the expected acceleration measurement from the angle from the total acceleration measurement. By approximating the K_v of the motor, we can Kalman filter the inertial acceleration and measured back emf of the motor. In theory, we could use the motor current command and measured resistance to accurately compute the back emf, but low–pass filtering the motor voltage also seems to work fine.

Before actually building an analog circuit to implement the control system, it was implemented digitally on the Arduino. That way, it was very easy to make changes to the control scheme without rewiring anything. When implemented in analog hardware, the control scheme that was tuned digitally worked quite well, with only a couple of tweaks needed.

7 Results

Overall, the cart performed extremely well, especially given that the only sensor onboard was an IMU. The motor controllers worked very well, rapidly and accurately changing the current through the motor without overheating the MOSFETs. The FETs in the motor driver were switching cleanly, without shoot–through current, and without any double switching. The Arduino Nano/DAC/gyroscope combination also worked satisfactorily, outputting multiple stable analog voltages corresponding to the IMU values it was designed to output with

low offset. The radio signal decoder also worked quite well. Testing showed that it would consistently output the neutral reference voltage of 2.5 V when it was fed a pulse width of 1500 µs, regardless of the frequency of the input signal. After low passing, the output was also pretty smooth. Mixing the radio receiver with the state variables allowed for a user to drive the segway around the lab.

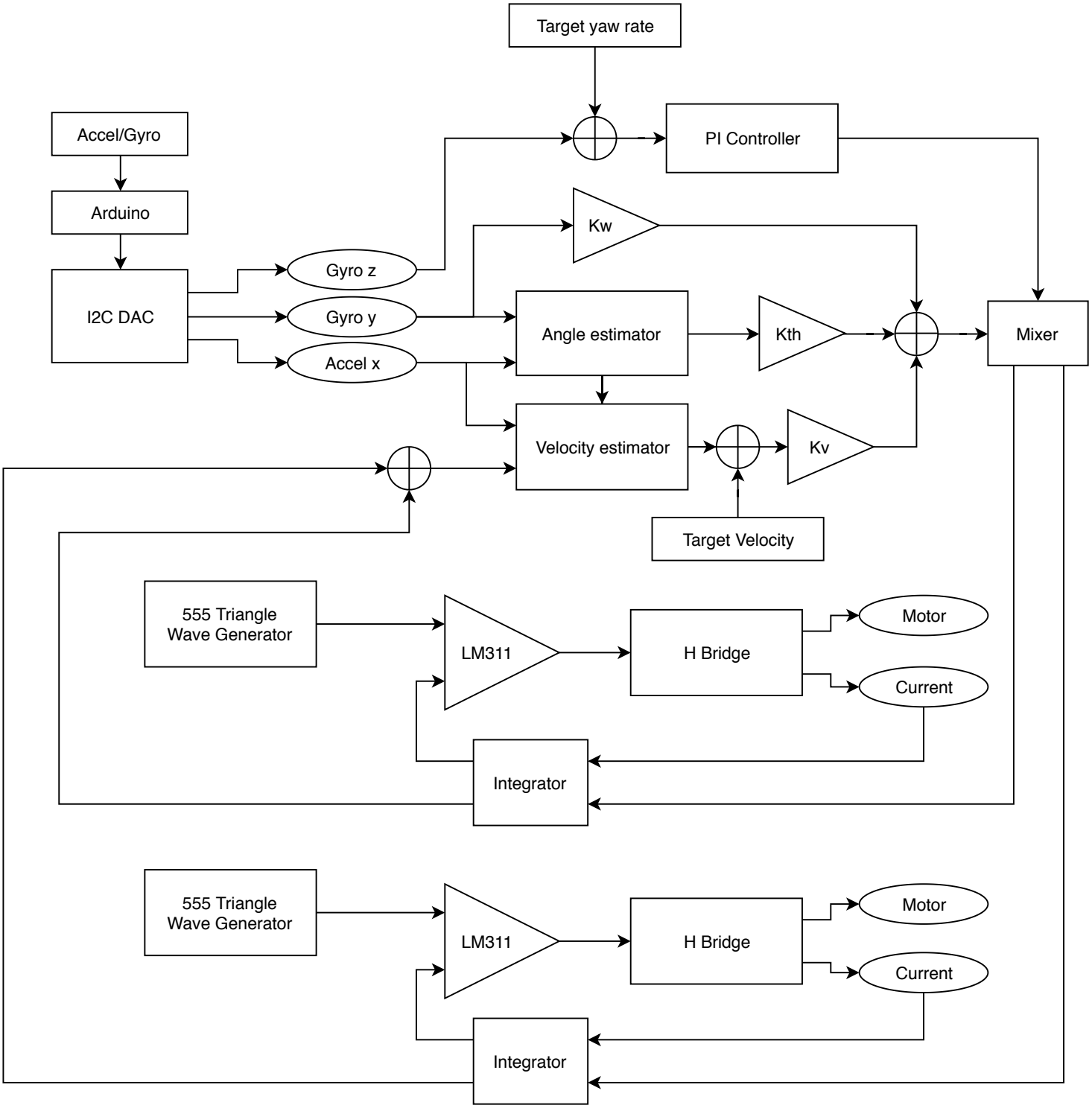
Despite the numerous approximations that went into designing the feedback controller, the estimators and controller performed very well. Rotating the cart around showed the the angle estimator quickly and accurately tracked the angle of the cart, and it was resistant to noise and errors caused by lateral acceleration. The velocity estimator also worked substantially better than simply reading the back emf voltage, although its design and performance could definitely be improved in future iterations of the segway.

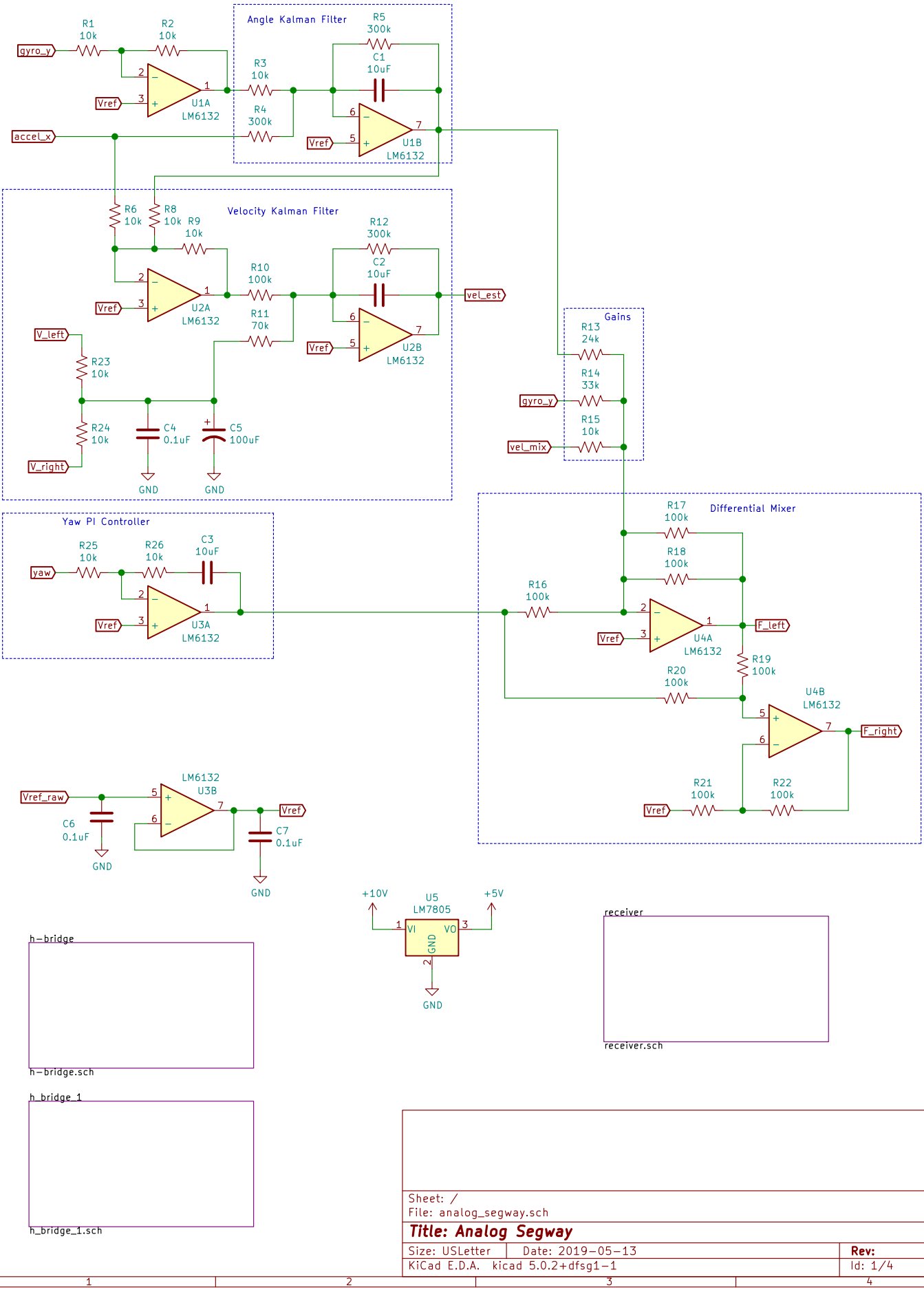
8 Conclusion

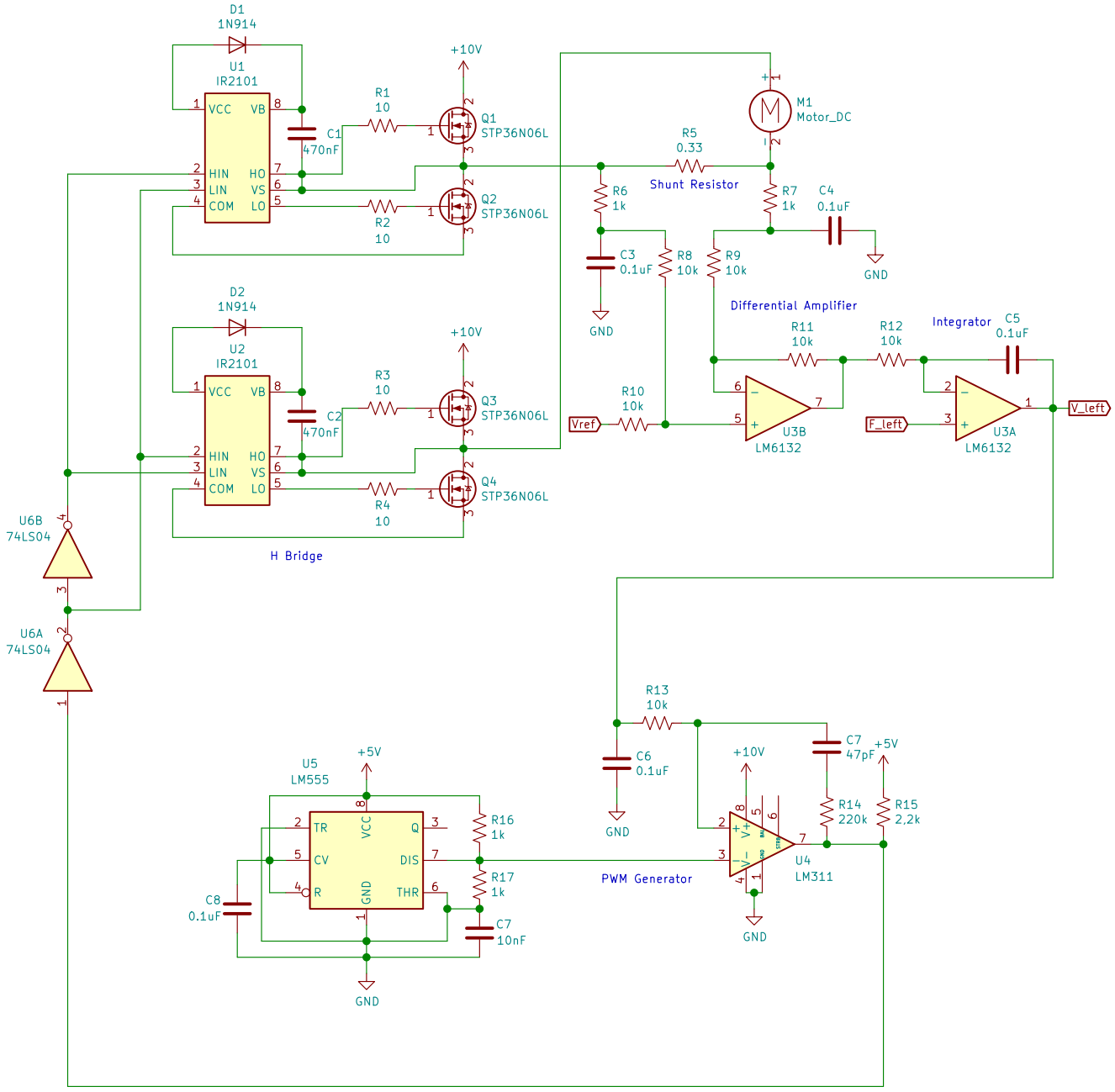
My goal for this project was to build a model inverted pendulum that was as completely analog as possible. I designed a current–controlled motor driver, a state–space feedback controller with estimators, and a circuit to decode servo signals to allow the servo to drive around. All of these components interfaced with each other through analog signals, resulting in a model segway that would stabilize itself and respond to commands from a radio receiver. Overall, this project was a very effective demonstration of the power of a well designed feedback system. The motor controllers were able to automatically adjust their PWM duty to match a target current, and the core feedback circuit consisted of multiple layers of feedback to internally model the system and use that model to stabilize the cart. Designing

the circuit required modelling and understanding how different components of the circuit would interact in order to choose gains that would result in a stable system. The segway also had to efficiently and rapidly regulate the current through a motor, as well as mitigating the noise from rapidly switching MOSFETs on and off. Finally, it had to reliably convert a discrete radio signal into an analog voltage, even when the timing of the pulses was uncertain. Building a fully analog segway was an enormous design challenge, but ultimately, extensive planning and many hours in lab resulted in a successful self–balancing cart.

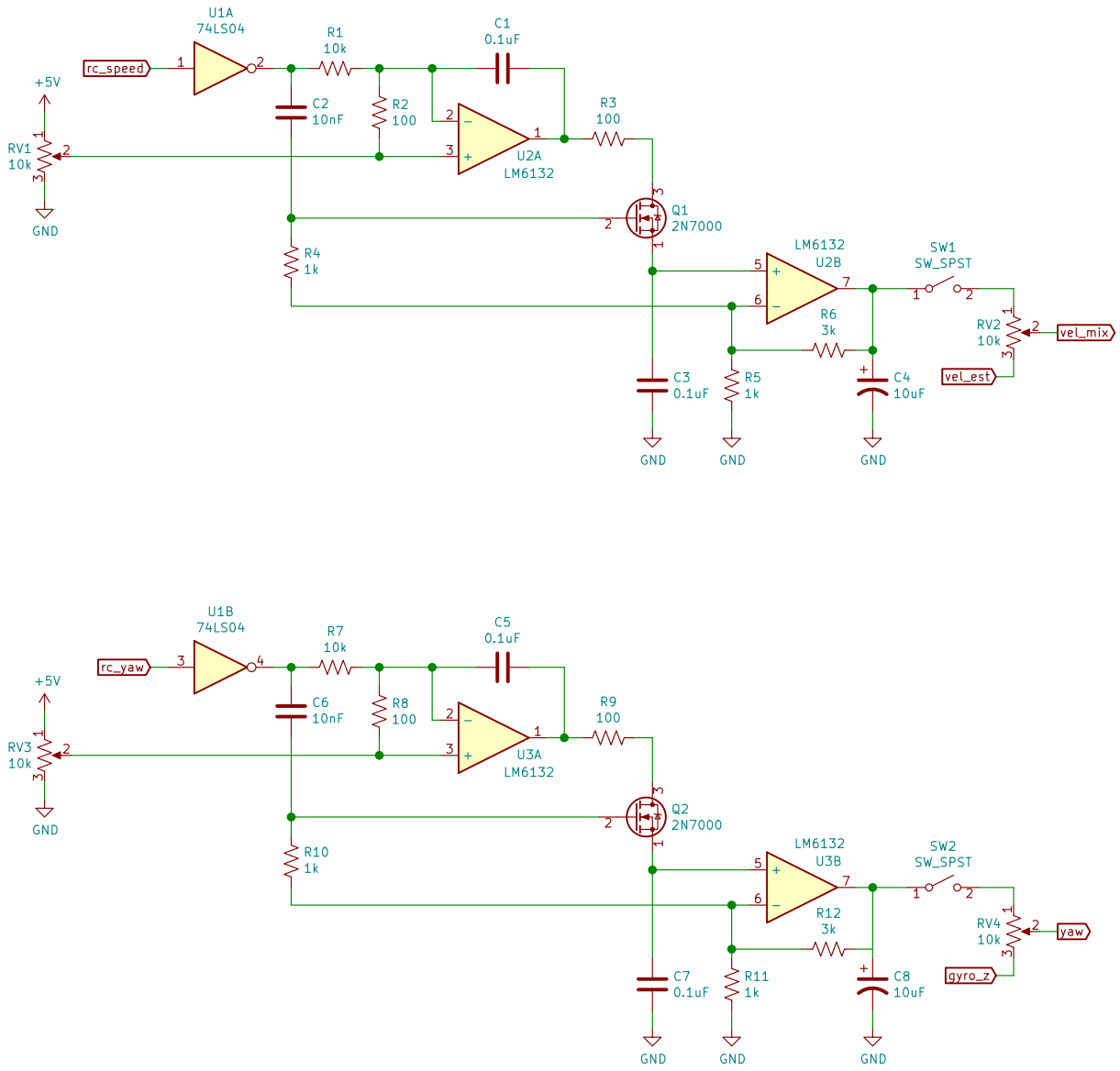
Simplified Block Diagram




Angle Kalman Filter
R1
10k
R2
10k
R3
10k
R4
300k
R5
300k
C1
10uF
gyro_y
accel_x
Vref
U1A
LM6132
U1B
LM6132
Velocity Kalman Filter
R6
10k
R8
10k
R9
10k
R10
100k
R11
70k
R12
300k
C2
10uF
U2A
LM6132
U2B
LM6132
vel_est
V_left
V_right
R23
10k
R24
10k
C4
0.1uF
C5
100uF
GND
Gains
R13
24k
R14
33k
R15
10k
vel_mix
Yaw PI Controller
R25
10k
R26
10k
C3
10uF
yaw
U3A
LM6132
Differential Mixer
R16
100k
R17
100k
R18
100k
R19
100k
R20
100k
R21
100k
R22
100k
U4A
LM6132
U4B
LM6132
F_left
F_right
Vref_raw
C6
0.1uF
U3B
LM6132
C7
0.1uF
+10V
U5
LM7805
+5V
VI
GND
VO
h-bridge
h-bridge.sch
h_bridge_1
h_bridge_1.sch
receiver
receiver.sch
Sheet: /
File: analog_segway.sch
Title: Analog Segway
Size: USLetter
Date: 2019-05-13
KiCad E.D.A. kicad 5.0.2+dfsg1-1
Rev:
Id: 1/4



Sheet: /h-bridge/		
File: h-bridge.sch		
Title: Left Current Controlled H Bridge		
Size: USLetter	Date: 2019-05-13	**Rev:**
KiCad E.D.A. kicad 5.0.2+dfsg1-1		Id: 2/4



Sheet: /receiver/ File: receiver.sch		
Title: Radio Receiver Decoder		
Size: USLetter	Date: 2019-05-13	**Rev:**
KiCad E.D.A. kicad 5.0.2+dfsg1-1		Id: 3/4

9 Gallery

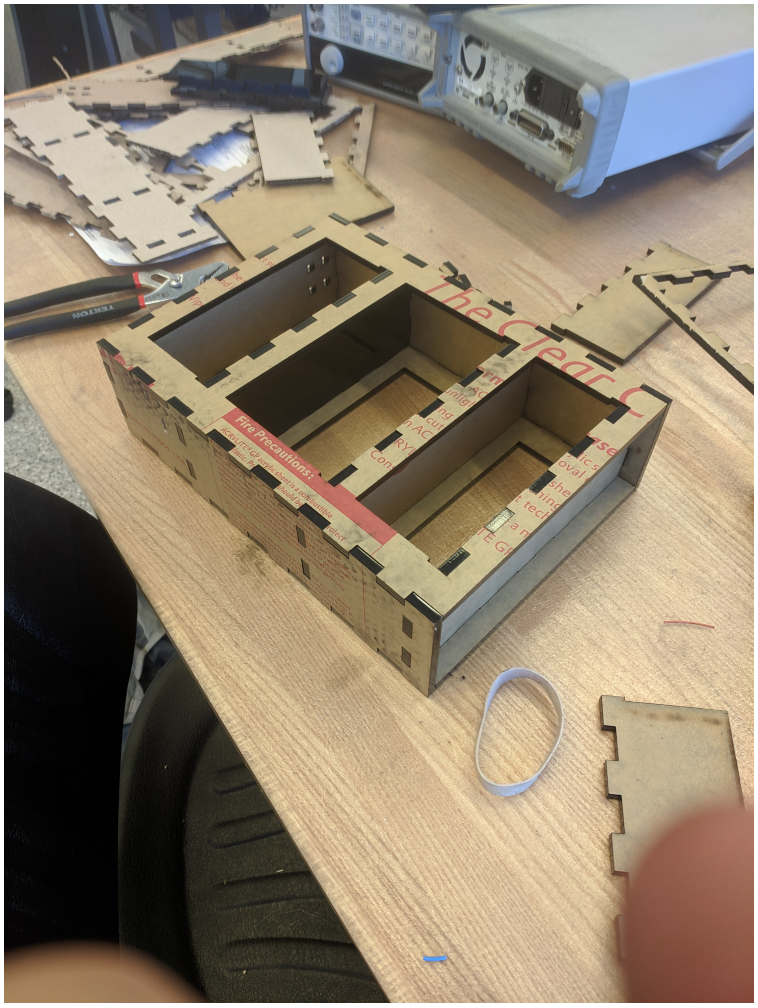


Figure 2: The frame before removing the protective backing from the acrylic and gluing it together.

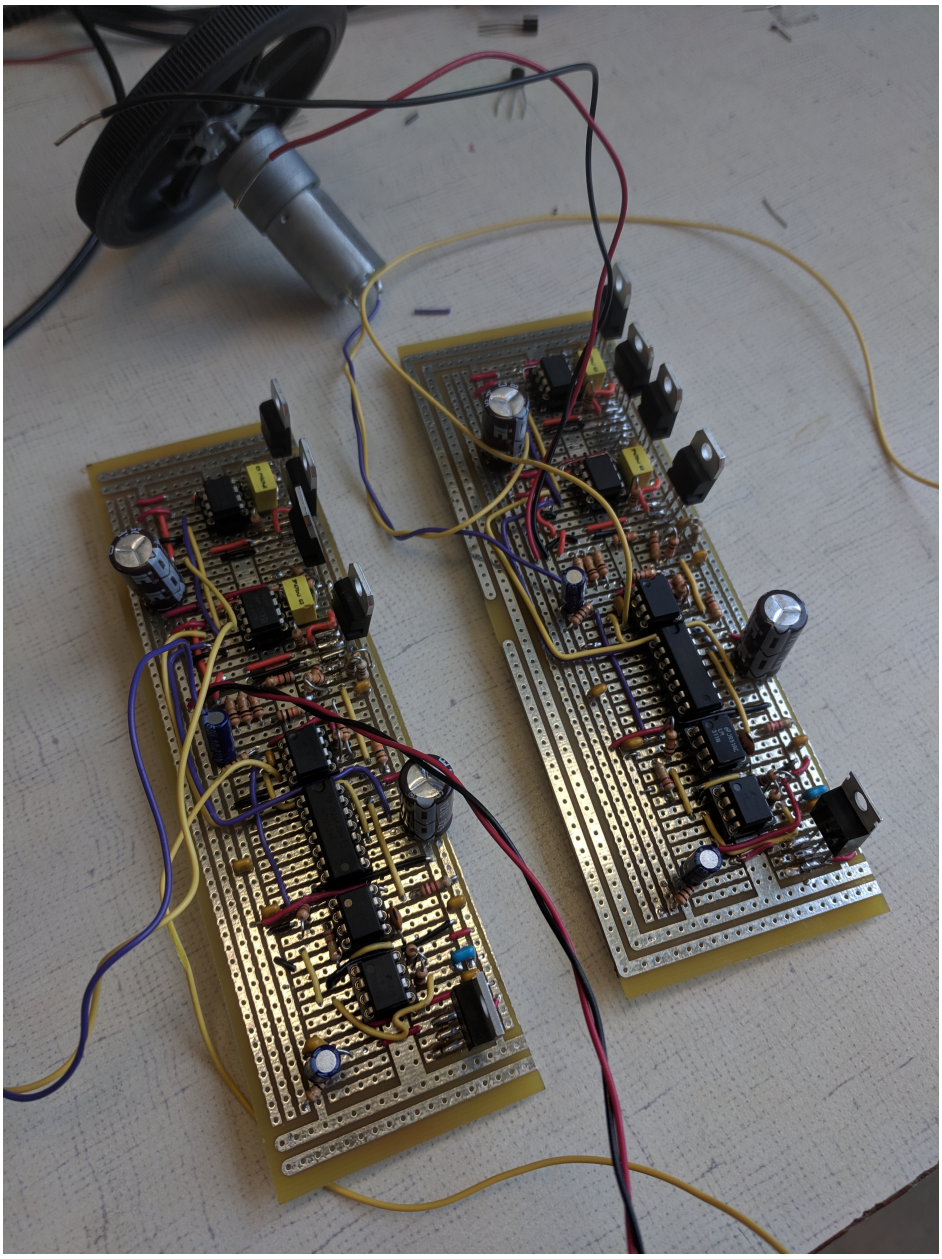


Figure 3: The motor drivers were soldered together on perf board.

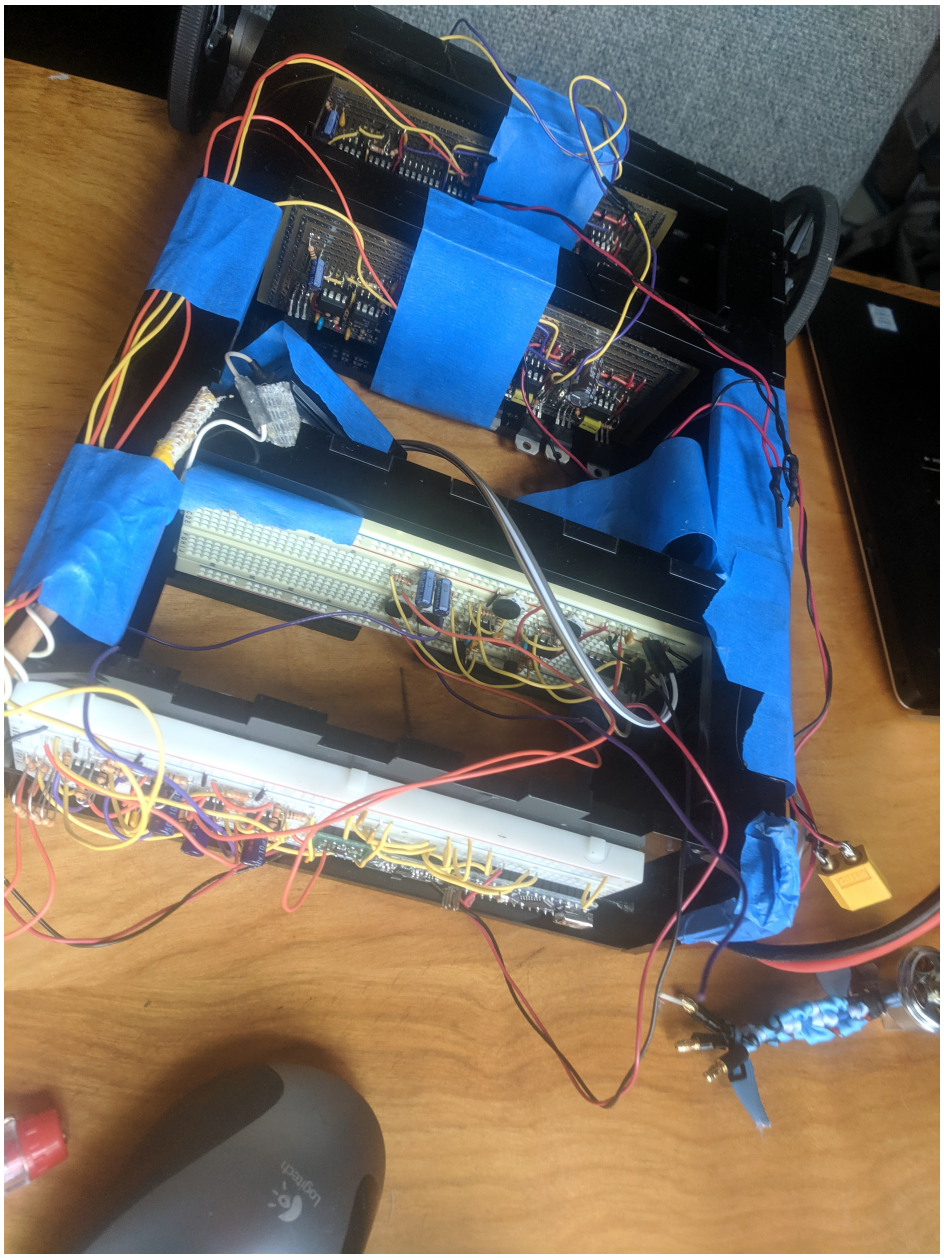


Figure 4: A picture of the completed segway. Many components are secured with masking tape. The top shelf has the state–space controller and the second shelf has the radio decoder circuit.

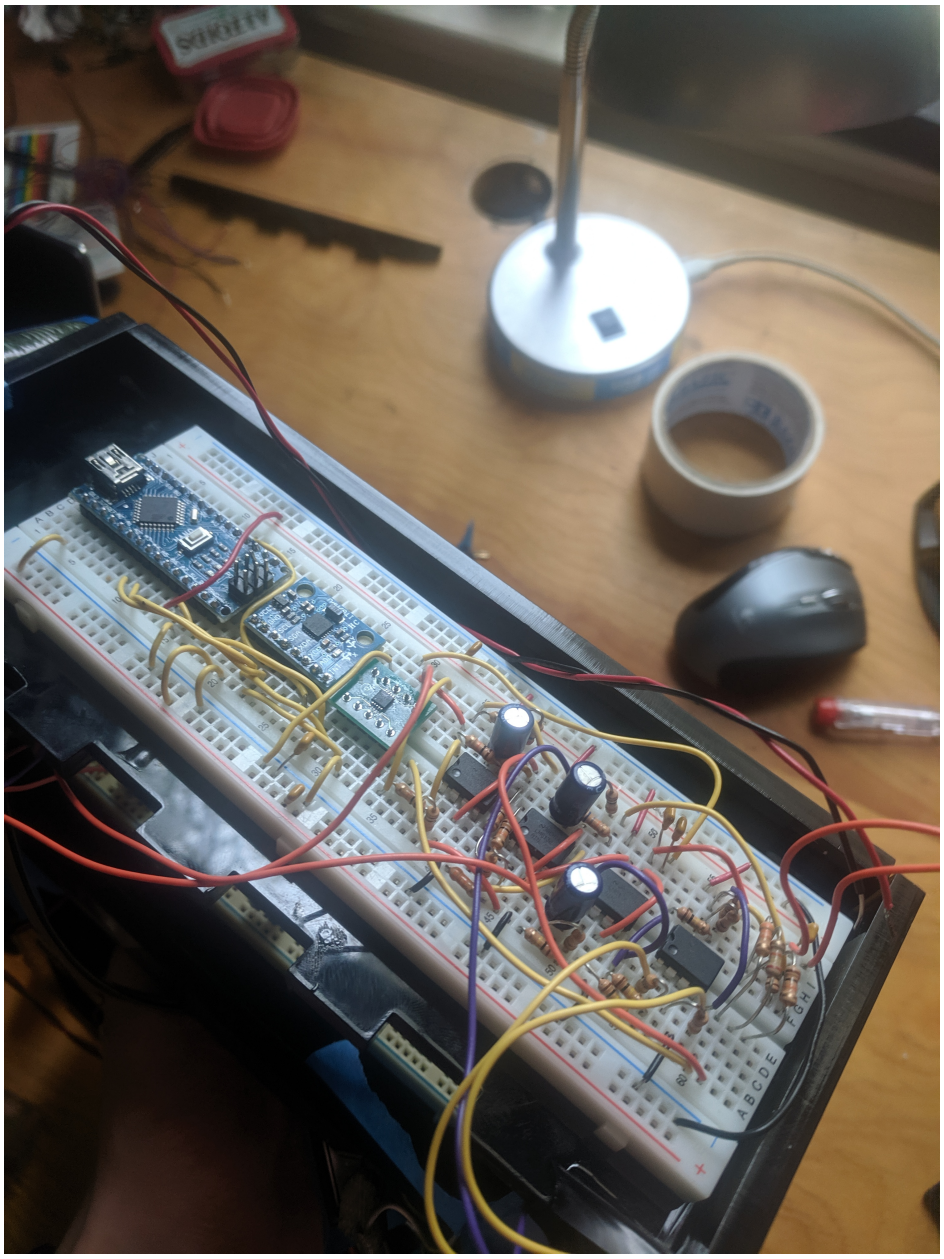


Figure 5: A more detailed image of the state–space controller circuit.

A Feedback controller derivation

Let θ be the angle of the cart measured from vertical, x be the cart position, and F be the horizontal force applied by the cart. We will define m to be the mass of the cart, g to be $9.8\,\mathrm{m/s}$, L to be the distance from the center of mass of the cart to the ground, I_{CM} to be the moment of inertia of the cart around its center of mass, and I to be the moment of inertia of the cart around the axle.

We can define the Lagrangian of the cart to be

$$\mathcal{L}(x, \dot{x}, \theta, \dot{\theta}) = T - U = \frac{1}{2} I\dot{\theta}^2 + \frac{1}{2} m\dot{x}^2 + m\dot{x}\dot{\theta}L\cos\theta - mgL\cos\theta$$

In addition, we have the constraints that

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}} - \frac{\partial \mathcal{L}}{\partial x} = F, \quad \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} - \frac{\partial \mathcal{L}}{\partial \theta} = 0$$

Evaluating these constraints gives

$$mgL\sin\theta = \ddot{\theta} I + m\ddot{x}L\cos\theta$$

$$m\ddot{x} + m\ddot{\theta}L\cos\theta - m\dot{\theta}^2 L\sin\theta = F$$

Linearizing these equations around $\theta = \dot{\theta} = 0$ gives

$$mgL\theta = \ddot{\theta} I + m\ddot{x}L$$

$$m\ddot{x} + m\ddot{\theta}L = F$$

With this information, we can make a state space model of the system that attempts to control velocity. We choose a state vector $\mathbf{q} = \begin{bmatrix}\theta,\ \dot{\theta},\ \dot{x}\end{bmatrix}^T$. This gives

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & I & mL \\ 0 & mL & m \end{pmatrix} \dot{\mathbf{q}} = \begin{pmatrix} 0 & 1 & 0 \\ mgL & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{q} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} F$$

Rearranging gives

$$\dot{\mathbf{q}} = \frac{1}{I - mL^2} \begin{pmatrix} 0 & 1 & 0 \\ mgL & 0 & 0 \\ -mgL^2 & 0 & 0 \end{pmatrix} \mathbf{q} + \frac{1}{I - mL^2} \begin{pmatrix} 0 \\ -L \\ I/m \end{pmatrix} F$$

Therefore, our open–loop system can be written as $\dot{\mathbf{q}} = \mathbf{A}\mathbf{q} + \mathbf{B}F$ for $\mathbf{A}, \mathbf{B}$ as defined above.

Let $\mathbf{p}$ be the setpoint of the system. Since we want the cart to be upright and not rotating, but moving with some velocity v, $p = [0, 0, v]^T$. If we let $F = \mathbf{K}(\mathbf{q} - \mathbf{p})$ for some $\mathbf{K}$, we have the closed loop system

$$\dot{\mathbf{q}} = (\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{q} - \mathbf{B}\mathbf{K}\mathbf{p}$$

Experimentally, if $m = L = 1$, $I = 2$, and $g = 9.8$, this system has eigenvalues near the negative real axis if $K = [20, 30, 4]$, giving a fast, clean response.