# 

# 

# **REMOTELY CONTROLLED**

# **SELF-BALANCING**

# **MINIBIKE**

By

Eric Tang

Jiaming Xu

Will Chen

Final Report for ECE445, Senior Design, Spring 2024

TA: Jason Zhang

1 May 2024

Project No. 22

**Abstract**

This report presents the design of a remote controlled self-balancing Minibike. Our project highlights the integration of multidisciplinary engineering principles to realize a sophisticated and efficient self-balancing minibike, demonstrating the synergy between mechanical design, electronics, and software engineering in a practical and innovative application. which allows the minibike to remain self-balancing with small disturbance from the environment and being remotely controlled at a considerable distance. This report shows the engineering process involved for this project, including the mechanical model, block diagram, detailed design processes, cost analysis, and performance analysis. Our project has almost met all the high-level requirements and achieved the functionality of the PCB successfully. Judging from our successful practice, the idea of allowing shared bicycles to find users on their own could become a reality in the future.

**Contents**

1. Introduction 5

1.1 Problem 5

1.2 Solution 5

1.3 Virtual Aid 6

1.4 High Level Requirements 6

2 Design 7

2.1 Block Diagram 7

2.2 Mechanical Design 8

2.3 Software Design 8

2.4 PCB Design 10

3. Design Verification 12

3.1 Mechanical Design Verification 12

3.2 Software Design Verification 14

3.3 PCB Design Verification 15

4. Costs and Schedule 17

4.1.1 Parts 17

4.1.2 Labor 17

[4.2.1](https://docs.google.com/document/d/1GnZulm_XumNYaaxJu25fSC5GeJhg_zaxYbCvtByRbSY/edit#heading=h.17dp8vu) Schedule 18

5. Conclusion 20

5.1 Accomplishments 20

5.2 Uncertainties 20

5.3 Ethical considerations 20

5.4 Future work 20

References 20

Appendix A Requirement and Verification Table 22

**1. Introduction**

**1.1 Problem**

Bike share and scooter share programs have surged in popularity across the globe in recent years. This mode of transportation is increasingly being recognized and supported for its convenience and environmental benefits. In Champaign, a company named Veo provides such services, offering a convenient option for short-distance travel between school buildings and bus stops. However, a notable problem is preventing more people from using this kind of transportation; these bikes can be parked randomly throughout the city. Inevitably, we will have bikes unevenly distributed around an area. The result is users often need to search for and walk to the nearest bike location, which can detract from the convenience of the service.

Several potential solutions to this issue have been considered, but each has its drawbacks. For instance, periodically collecting and redistributing all of the bikes could address the problem of uneven bike distribution. However, this approach is likely to be costly and inefficient, requiring significant logistical effort and resources. Another potential solution involves flooding the region with a high number of bikes to ensure availability in most areas. Yet, this method raises concerns about cost efficiency and potentially cluttering public spaces with excessive bikes and scooters.

**1.2 Solution**

To solve this problem practically, we need tracking and management systems for bikes and scooters, that encourage users to redistribute bikes. But more importantly, use the tracking and management systems to deliver the bikes to users. To achieve this, we first need to have a self-balancing Bike. This is the goal of our project. First, we will construct a 40 cm-long bike model, incorporating a gyroscope, two 11V motors, one 3.3V motor, a flywheel, a control system, and an 11.1V LiPo battery. Second, we will utilize data from the gyroscope along with the velocity feedback from the BLDC motors to adjust the torque on the flywheel, ensuring the bike remains balanced. Third, we will use one of the BLDC motors to propel the rear wheel, enabling the bike to drive straight, while the steering motor will be tasked with controlling the direction. Finally, we will connect the bike's control to an external controller, enabling the bike's remote operation.

**1.3 Virtual Aid**

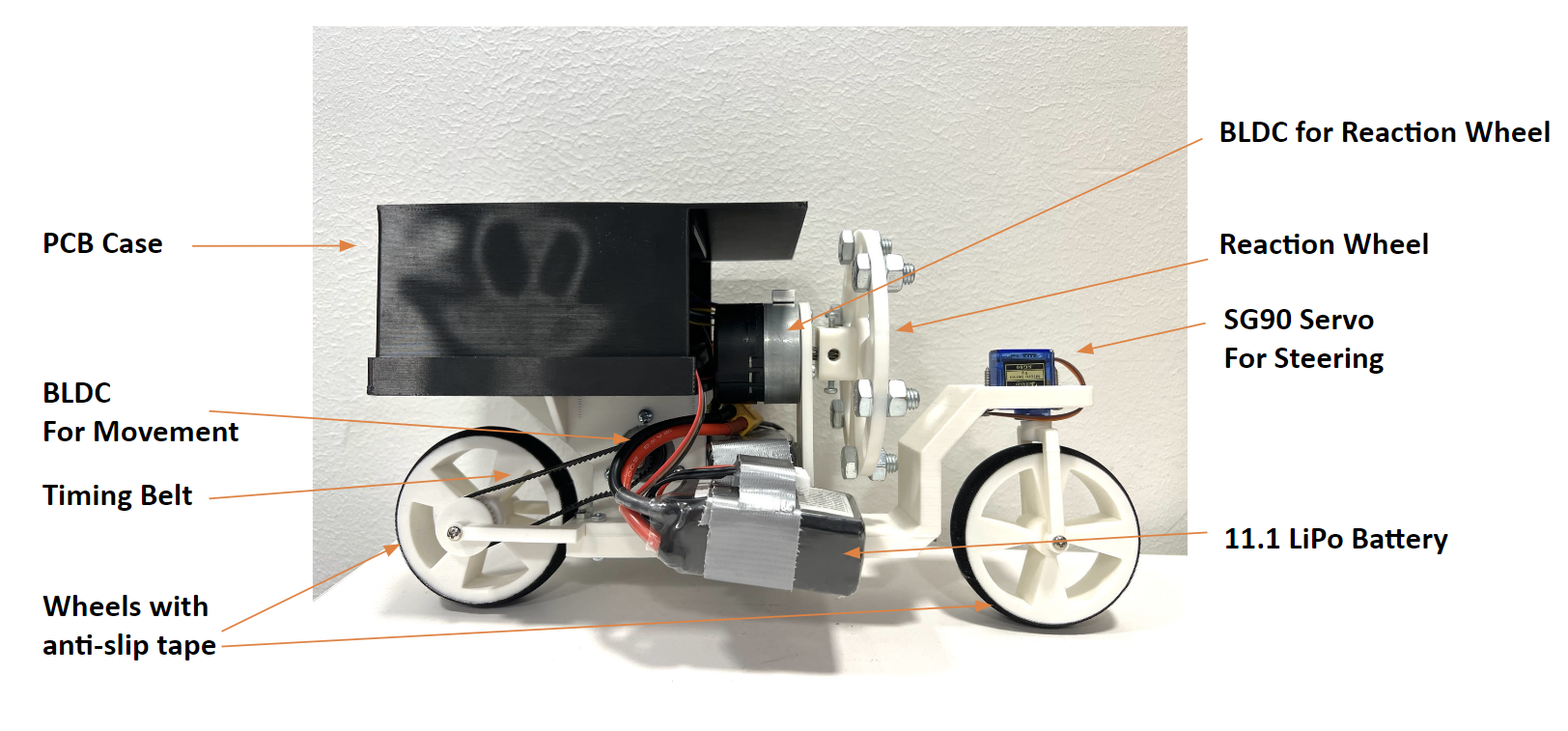


Figure 1: Visual Aid

**1.4 High-Level Requirements**

The bicycle should be able to remain self-balanced when the power is on.

The bicycle should be able to self-regulate lateral force disturbances within +- 10 degrees and continue to maintain balance.

The bicycle should be remotely controlled by the controller within a radius of 30 m from a bicycle.

**2 Design**

**2.1 Block Diagram**

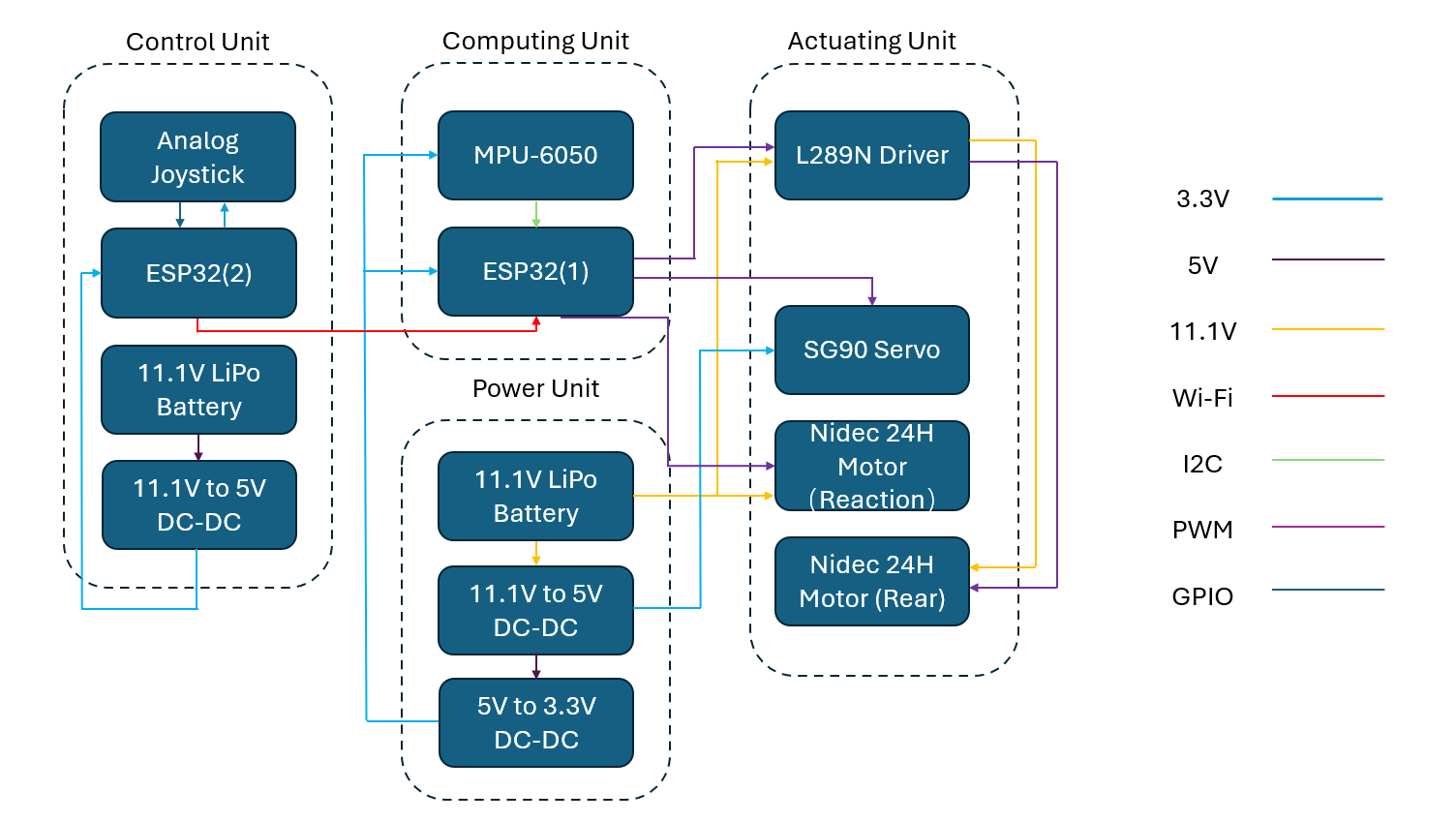
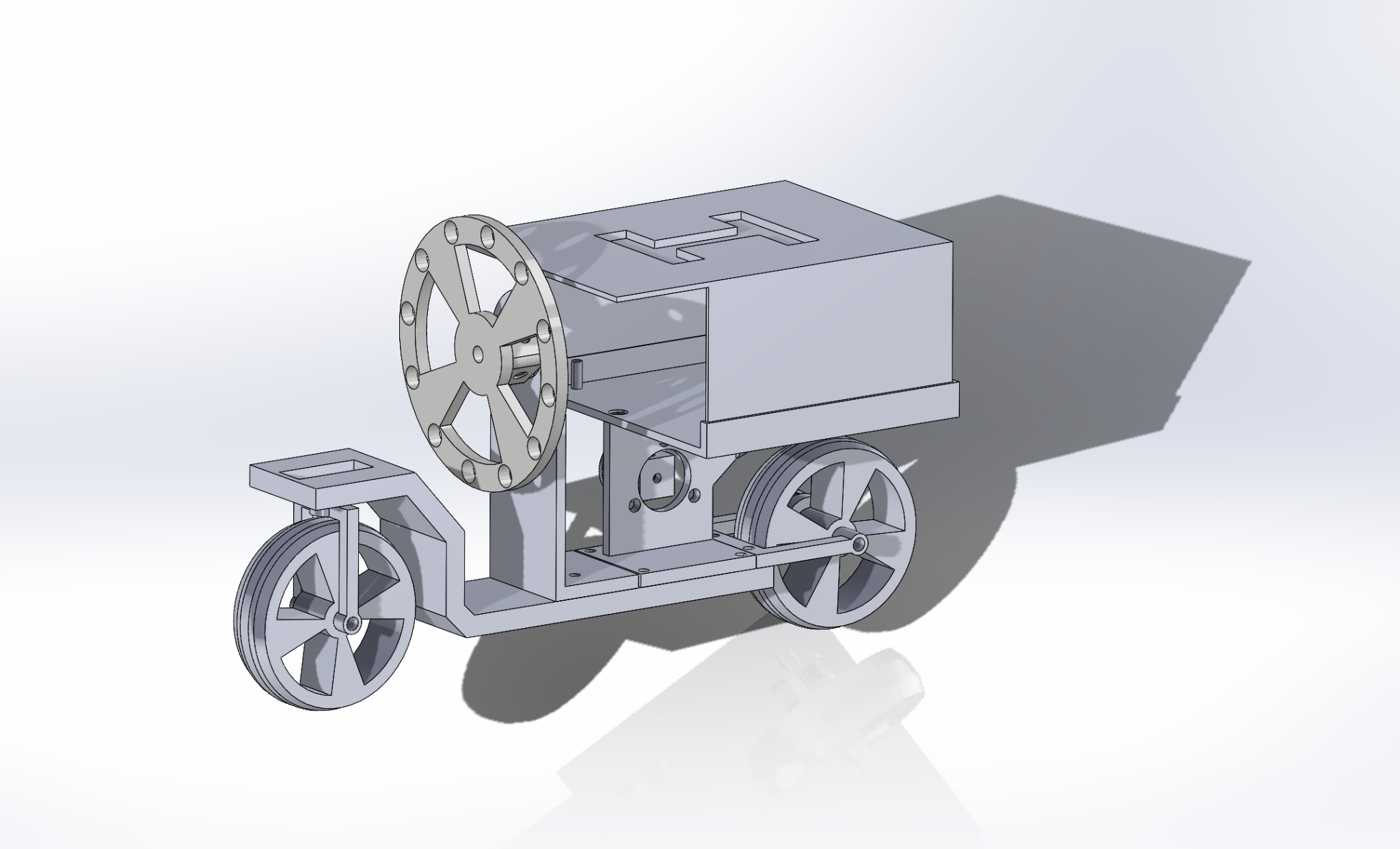
****

Figure 2: Block Diagram

**2.2 Mechanical Design**

**2.2.1 Bike Physical Design**

**** Figure 3: 3D Model in Solidworks

We designed the minibike frame with SolidWorks, and we plan to print the frame out with PETG filament and a 3D printer. All electronics and two wheels will be installed to predefined positions on the bike frame with screws and zip ties.

**2.3 Software Design**

**2.3.1 Control Subsystem**

We plan to design a Wi-Fi 2.4GHz remote controller using two ESP32-S3-WROOM microcontrollers and a two-axis pushbutton rocker sensor module by joystick potentiometer to implement the function of remote control. One ESP32-S3-WROOM serves as the transmitter which functions as the primary input source, providing control inputs such as acceleration, braking, and steering. It communicates these inputs wirelessly to another ESP32-S3-WROOM microcontroller function as a receiver. The ESP32-S3-WROOM Microcontroller decodes and processes these data packets to extract control commands, including steering angle, brake intensity, and any additional functions mapped to the controller buttons. The communication between two ESP32 microcontrollers will be coded based on ESP\_NOW, which is a wireless peer-to-peer Wi-Fi communication protocol. ESP-NOW utilizes Wi-Fi technology, which typically offers better range and stability compared to Bluetooth or traditional RF protocols. It also supports higher data rates compared to Bluetooth and RF protocols, allowing for faster transmission of control commands and data between the remote controller and the minibike, which is beneficial for our project since it requires real-time control.

**2.3.2 Actuating Subsystem**

The Actuating Subsystem consists of two 11V Brushless DC motors and a 5V servo motor. One of the 11V electric motors is used for propelling the bike by utilizing a chain/belt system that connects to the rear wheel. The other 11V motor is used for powering the flywheel to input a new torque into the bike system and balance the bike when it is leaning. The 5V servo is used to steer the front wheel. The motor control needs to control the high-power motor to do quick input changes. It needs to change the torque exerted by one electric motor onto the flywheel in a very short amount of time.

**2.3.3 Computing Subsystem**

The Computing Subsystem consists of an MPU6050 accelerometer and an ESP32 S3 WROOM microcontroller. The MPU6050 will take measurements and feed that information into the microcontroller; then, the microcontroller will process this information and calculate the tilt angle of the bike and appropriate input to the motor that connects to the reaction wheel.

**2.3.4 PID Control**

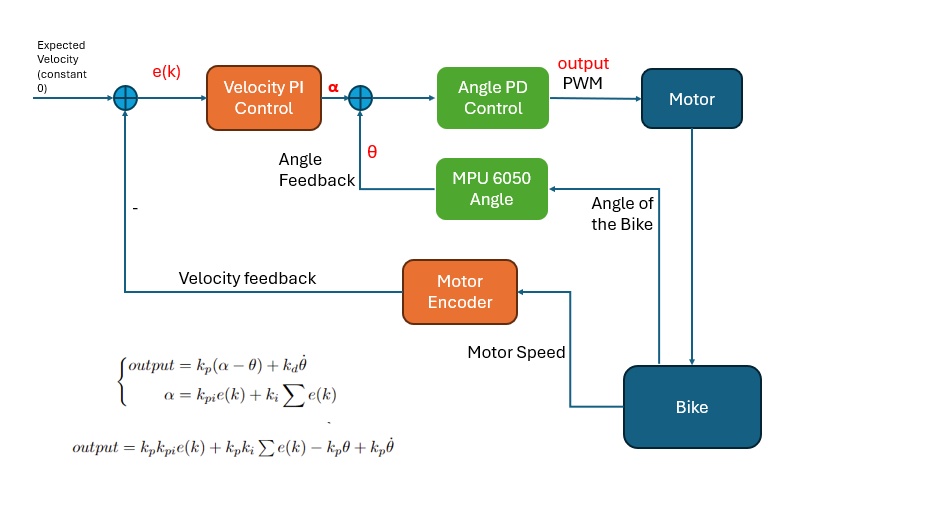


Figure 4: PID Control

For our control logic, we implemented a nested closed loop control, a closed loop control based on velocity and a closed loop control based on angle.

For the Vertical Control Loop, we first read the raw reading from MPU 6050 and then calculated the tilted angle of the bike and the angular velocity of the bike. Then, we applied a Kalman Filter to filter out the unnecessary noises from the sensor and subtract the initial biases from the raw reading. We finally feed the filtered angle reading and the filtered angular velocity reading into the Vertical Loop with corresponding gains, Vertical\_Kp and Vertical\_Kd.

For the Velocity Control Loop, we first read the pulse signals from the rotary encoder of the Brushless DC Motor and then calculated the corresponding speed of the motor. We continuously added up the motor speed readings to get the position, and we constrained the position with +-110. We finally feed the motor speed reading and the position reading to the Velocity Look with corresponding gains, Velocity\_Kp and Velocity\_Ki.

The Vertical loop expects the angle of the bike to be zero and will try to keep the bike balancing, and the Velocity loop expects the motor speed to be zero and will try to make the motor stop when the bike is balanced.

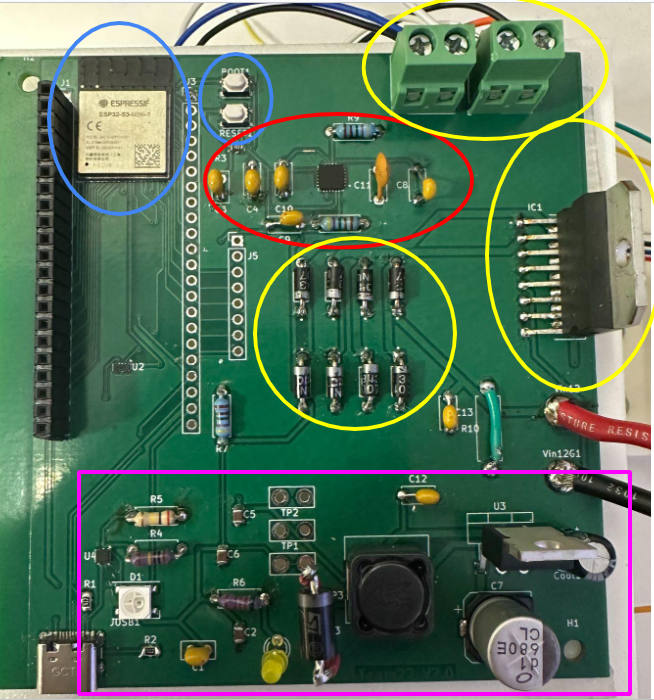
**2.4 PCB Design** 

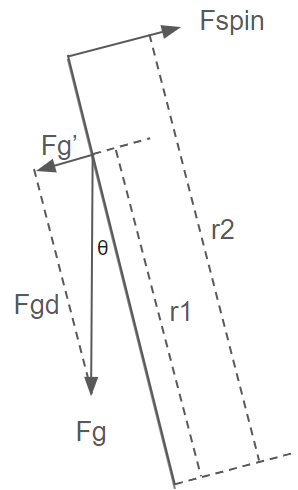
Figure 5: PCB

Our PCB has four major components. As shown at right. These are Microcontroller ESP32(blue), Accelerometer MPU6050(red), Motor Control L298N(yellow), and Power system(consists of a 12V to 5V buck converter and 5V to 3.3V linear converter, all are pink). We locate ESP32 at the edge of the board for less antenna area noise. We put L298N in a relatively empty area and at the edge of the board due to a forum search informing us that L298N might dissipate a lot of heat. Screw terminals and type C connectors are all at the edge for better connections. We also left three test points for our power system to test and substitute our power system in case one of the converters isn’t functioning as expected. We are able to use a test bench to drive the circuit board at a preferred voltage. We also connect every pin of ESP32 to a pinout. This way we can test pins each by each. This will facilitate debugging greatly since we can try different ports on the ESP32 board to narrow down the issue.

The power subsystem of our project serves as the backbone, ensuring the stable and reliable operation of all components. At its core is an 11.1V LiPo battery, strategically chosen for its capacity and voltage output. This battery not only powers the entirety of our project but also directly supplies energy to drive DC motors, essential for the propulsion of our mini car. To cater to the diverse power requirements of various components, we've implemented two voltage regulators. The first, an 11.1V to 5V buck regulator, efficiently steps down the battery voltage to a stable 5V output. We plan to use LM2596 for this regulator. This regulated 5V supply is dedicated to powering the SG90 Servo motor, ensuring consistent torque delivery for precise control. The second regulator, a 5V to 3V linear voltage regulator, further refines the power output, providing a clean and stable voltage source for delicate components such as the ESP32s and MPU6050. With this meticulously designed power subsystem, we ensure optimal performance, longevity, and reliability across all facets of our mini-car project.

**3. Design Verification**

**3.1 Mechanical Design Verification**

Left is a brief force analysis of the bike model when tilted at angle θ

r1: center of the mass to the ground

r2: reaction wheel connecting point to the ground

Fg: gravity pull by the mass of the bike

Fg’: component of Fg parallel to Fspin

Fgd: component of Fg, combined with Fg’ we have Fg

Fspin: created by the torque of the motor when powering the reaction wheel

θ: tilting angle of the bike model, max is 10 degrees in our design

Figure 6: Force Analysis

We start with getting the equilibrium condition of the above situation, and then utilizing that we derive an equation that relates torque to mass and r2.

;

To get equilibrium, we need two torque to be the same:

m here is the total weight of the whole bike, which consists of the weight of the reaction wheel, motor, the bike itself, wheels, etc.

In short, we want the reaction to have a low weight and high inertia. High inertia will decrease the angular acceleration of the reaction wheel, thus keeping the rpm of the motor low which is easier for tuning and has a bigger compliance range since the motor has a maximum spinning speed. The low weight helps to keep the center of mass low and keep the total mass down. To reach such a design goal, we designed the reaction wheel like a ring. We first assumed the weight of the reaction wheel and proceeded to calculate and verify.

We also need to calculate the center of the mass of the bike model. We assume we can treat the bike’s center of gravity in a 1D line closer to the reaction wheel. Since the distance between each heavy object is small, thus we can ignore the imbalance along the bike.

=

Thus we got the equation we need to determine the torque. It should be a pretty small value due to our bike being very small.

Now combine the equation we had previously and weight of parts we calculate the desired torque.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Obj. description | weight(g) | distance(cm) |
| m1 | reaction wheel motor and reaction wheel | 355 | 12.5 |
| m2 | driving motor | 215 | 6.5 |
| m3 | battery | 153\*2 | 4.5 |
| m4 | bike model | 235 | 6.2 |
| m5 | servo motor | 10g | 9.5 |

m = 1121g = 1.121kg; r1 = 7.8cm = 7.8\*10-3m

= = 1.121 \* 9.81 \* sin(10) \* 7.8\*10-3= 0.015 NM

Combined with all of the above results. We calculated our minimum torque required is 0.015 NM, and our motor’s maximum torque is 0.5NM. Which is very sufficient for this project.

**3.2 Software verification:**

There are three main components for the software design:

**3.2.1 Gyroscope Reading**

There are two main requirements for the full functionality:

R1. The microcontroller can interpret the correct tilt angle based on measurement.

Verification 1: After implementing the script of converting the raw data from MPU6050 to the tilt angle we need, we fixed the gyroscope with several fixed angles: 0, 30, 60, and 90 degrees, and see whether or not the angle we read is the same as the angle we expected. After multiple tests, we concluded that the angle reading has the right changing tendency and also matches our angle expectations.

R2. The noise of the readings is within acceptable range (+-0.1 degrees)

Verification 2:

To verify our results, we use the serial plotter in Arduino IDE to see how large the noise is. At the beginning, we realized that the gyroscope’s reading is pretty sensitive to environmental changes. For example, even the moving air would cause the readings to fluctuate greatly. To solve the large noise, we applied the Kalman filter. Then we use the serial plotter to observe the output before and after the Kalman filter. It turns out the tolerance to the environment noise increases a lot, and the noise falls within an acceptable range.

**3.2.2 Remote Control**

R1. The remote control must maintain stable communication with the bicycle with a delay <= 20ms

Verification 1:

To verify this part, we use the timestamp to observe the time interval among the commands. We keep sending different movement commands like turn left 30 degrees or turn right 60 degrees and observe the timestamps. The result is that the delay for each pair of commands is within 10ms.

R2. The control signal receiver can receive the signal from the transmitter successfully.

Verification 2:

The verification of this part is done by observing the error build-in error messages and then comparing the messages one chip sends is the same as the message another chip received. To check the error messages, after we set up the protocol, we check potential errors like initialization failed or peer not found, and other potential errors in sequential order. Once there’s no error showing in the terminal, we tried to send the different commands from one chip to another and connected both chips to two serial monitors to print out the commands. The result is that two chips can communicate successfully and the command we send is the same as the command we received.

R3. The remote controller should be able to control the mini bike steadily within a range of 30 meters

Verification 3:

To verify this, one of our team members held the remote controller on one end of the second floor of ECEB, and we put the bike on another end of the hallway. The bike can still be remotely controlled successfully. The distance between the two ends is approximately 70 meters, which is far more than the 30 meters in the requirement.

**3.3 PCB Design Verification**

To verify our PCB, we first start with the power system. We connect the input with a 12V test bench and test 5V and 3.3V test points with a multimeter. During the process we observe the input current on the test bench closely. If any voltage is not producing correctly, we can use the test-bench to avoid the potential faulty system and test the other power system first. After that we rule out each possibility one by one, from potentially bad soldering, wrong components to faulty components and faulty PCB boards.

After we verify the Power system working properly. We verify the ESP32 is functioning properly by first trying to connect the chip to the computer. If it is not connected we verify the input voltage, enable voltage and soldering. After that we check the equipment around it. After that we try to write code into it, letting it set one of its nodes' voltage to high, then let it send a square function. Then we verify it receives the MPU6050 properly by first ensuring powering nodes for mpu6050 delivers a consistent 3.3V. Then we let ESP32 connect to MPU6050, after that we try to receive the signal from MPU6050 and print it out on the laptop.

Our battery is an 11.1V 1500mAh Lipo battery. We can convert it to Wh which is：

11.1V \*1.5Ah = 16.65 Wh

|  |  |  |
| --- | --- | --- |
| Part | Average Current Draw @ 5 V | Total Current:  0.418A |
|
| Processor | 150mA |
|
| Motor Controller | 50mA |
| To be safe we give an extra 20% of the designed current needed. Thus, we will design with a 0.5A current draw from 5V system |
| Servo Motor | Average of 300mA when active, and 10mA when idle, we assume the Servo needs to turn the wheel around 70% of the time thus the average current draw is:  300mA \* 70% + 10mA \* 30% = 213mA |
|
| accelerometer | 5mA |
|

The 12V to 5V converter system is rated to 3A output, and the 5V to 3.3V system is rated at 1A. Which are more than sufficient to power the system.

For our 5v system with an efficiency(eff) of 85%, thus it will have an average power of

Pavg = Vout \* Iavg / eff = 5V \* 0.5A / 85% = 2.95W

For the two 11V electric motors, the propelling wheel and flywheel should both have a relatively stable operating environment, thus I assume in general 15% of the time running at maximum power and 85% running under normal operating current(0.7A). This means

Pavg = V11 \* Iavg = 11V \* 2 \* (0.7A \* 85% + 1.8A \* 15%) = 19W

Thus, for normal operation the system has a net power consumption of 21.95W, which means under normal operating circumstances the bike can run 45 min on a single battery.

For extreme cases, we assume that the propelling wheel is running at maximum torque, and the flywheel motor runs at maximum torque for 50% of the time.

Pavg = V11 \* Iavg = 11V \* ((0.7A \* 50% + 1.8A \* 50%)+(1.8\*100%)) = 33.55W

Thus, our total power consumption of the system is 36.5W, which means the system can run on a single battery for 30 minutes.

# 

# **4. Cost and Schedule**

**4.1 Cost**

**4.1.1 Labor:**

We assume that the average salary of an ECE graduate is about $50 per hour. This project is composed of 3 members, and we contribute roughly 14 hours per week. We estimate that this project will take about 10 weeks to complete.

Thus, the total labor cost = $50 \* 3 \* 14 \* 10 = $21,000.

**4.1.2 Parts**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Manufacturer | Quantity | Unit Price | Cost | Description |
| LM2596 | Texas Instrument | 1 | $4.50 | $4.50 | 11V to 5V DC to DC |
| L298N | DFRobot | 2 | $12.9 | $25.8 | Motor Driver |
| SG90 | Beffkkip | 1 | $7.99 | $7.99 | Servo Motor |
| MPU 6050 | Foriot | 1 | $8.69 | $8.69 | Accelerometer |
| Wheel | Uxcell | 2 | $8.00 | $16.00 | Wheels |
| LiPo 11.1V | Ovonic | 1 | $16.50 | $16.50 | Battery |
| PETG Filament | Overture | 1 | $16.14 | $16.14 | Filament for 3D Print |
| Joystick | HiLetgo | 1 | $6.29 | $6.29 | Joystick control |
| BLDC 24H | Nidec | 2 | $5.90 | $11.80 | BLDC motor |
| Circuit elements | N/A | N/A | $15.00 | $15.00 | circuit elements |
| ESP32-S3-WROOM | HiLetgo | 2 | $9.99 | $19.98 | Microcontroller |

**4.1.3 Total Costs**

Our estimated labor costs will be $21,000, and our estimated cost of all parts will be $148.69. Thus, our total cost will be $21,148.69.

**4.2 Schedule**

|  |  |  |
| --- | --- | --- |
| Week | Task | Team Member |
| Week-01 | Brainstorming | All |
| Week-02 | Brainstorming  CAD Assignment | All  All |
| Week-03 | Brainstorming  Proposal | All  All |
| Week-04 | Researching other plausible solutions  Team Contract | All  All |
| Week-05 | Researching necessary parts and electronics | All |
| Week-06 | Design Document | All |
| Week-07 | Design Review  Bike Frame prototyping  PCB Schematic Design | All  Jiaming  All |
| Week-08 | PCB Design  Motor Testing  Remote Controller | Eric  Jiaming  Will |
| Week-09 | N/A (Spring Break) | N/A |
| Week-10 | PCB Refining  Self-balancing tuning  Remote Controller Testing | Eric  Jiaming  Will |
| Week-11 | PCB Refining  Bike Movement | Eric  Jiaming & Will |
| Week-12 | Test and refine | All |
| Week-13 | Final Testing  Mock Demo | All  All |
| Week-14 | Final Demo and mock presentation | All |
| Week-15 | Final Presentation | All |

**5. Conclusion**

**5.1 Accomplishment**

In conclusion, we successfully made the bike self-balancing given small external disturbance, and the bike could be remotely controlled with a range of at least 70 meters. All subsystems of our PCB work, and it is almost fully functional.

**5.2 Uncertainties**

However, there is one existing uncertainty or failure in our project.

First of all, the microcontroller, ESP32, on the PCB can not read accurate motor feedback from the motor encoder. We have tried all the pins on ESP32, and they all failed. We concluded that the failure was due to interference from other components on the PCB, and one possible solution would be to isolate sensitive components and apply proper routing techniques.

**5.3 Future Work / Alternatives**

As for future work, we plan to add an additional sensing mechanism (LiDAR or RGB D Camera) with a particle filter like Monte Carlo Localization[1] to localize the bike’s position and pose, and then achieve autonomous driving features. In addition, we plan to use Deep Reinforcement Learning Control[2] to make the self-balancing bike more robust to handle larger disturbances.

**5.4 Ethical Considerations**

Throughout this project, we adhere to the IEEE Code of Ethics. We will “uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities”[3]. We will “treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others”[3]. We will “strive to ensure this code is upheld by colleagues and co-workers”[3].

The Lithium battery must be kept safely inside a fire-retardant charging bag for storage and charging. For storage a few days long, we need to find a dry and cool place that won’t be directly hit by sunlight. For storage longer than a few days, we need to follow all the instructions above and use a meter to measure the voltage and make sure their voltage is around 4.4V. When charging the battery, we can only use the provided charger from the manufacturer. During usage, we will charge the 11V battery frequently to ensure the power of our 11V motors.[4]We will do voltage testing every time after charging the battery and document the voltage, and visual inspection every time before and after charging the battery. When batteries are not in use we need to disconnect them from the cart and store it in a dry enclosed transparent box. If the battery shows any signs of leakage or malfunction, we must immediately stop using it and depose it appropriately.

**Reference**

[1] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo localization for mobile robots," Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 1999, pp. 1322-1328 vol.2, doi: 10.1109/ROBOT.1999.772544.

[2] Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, Ivana Palunko,

Reinforcement learning for control: Performance, stability, and deep approximators, Annual Reviews in Control, Volume 46, 2018, Pages 8-28,ISSN 1367-5788, https://doi.org/10.1016/j.arcontrol.2018.09.005.

[3] “IEEE Code of Ethics.” IEEE, https://www.ieee.org/about/corporate/governance/p7-8.html

[4] “Battery Safety” UIUC Division of Research Safety, <https://drs.illinois.edu/Page/SafetyLibrary/BatterySafety>

Appendix 1: Requirements and Verification Table

**Control Subsystem**

|  |  |
| --- | --- |
| **Requirement** | Verification |
| **R1. The remote control must maintain stable communication with the bicycle with a delay <=20ms** | V1. We can add a timestamp when the microprocessor receives the data from the remote controller. The time delay is the time difference between time stamps. We can let the remote controller send the signals to the microprocessor continuously. By tracking the time intervals for each signal sending, we can check whether or not the delay is <= 20ms |
| **R2. The control signal receiver can receive the signal from the transmitter successfully** | V2. We will add print statements on the microcontroller's script to check whether the signal that the transmitter is sending out is the same as the signal the receiver receives. The print statements can be observed on the computer screen directly |
| **R3. The remote controller should be able to control the mini bike steadily within a range of 30 meters** | V3. We first measured the distance of 50m with a soft ruler, and then moved away from the bicycle in units of 10m with the bicycle as the center of the circle and used the remote controller to send forward, backward, and steering commands to observe whether it was successful and delayed within the acceptance range |

**Actuating Subsystem**

|  |  |
| --- | --- |
| **Requirements** | **Verification** |
| R1. The reaction wheel needs to be balanced to spin at a high rpm | V1. Attach the motor with the reaction wheel and start to spin the reaction wheel when attached to a stick. Watch the reaction wheel and motor’s vibration, if the stick shakes badly we need to rebalance the reaction wheel, or we have to lower the rpm for more stable balancing components. |
| R2. The reaction wheel motor is strong and precise enough to mitigate torque | V2. Attach the motor with the reaction wheel on a torque measuring stick, and let the motor spin the reaction wheel while securing the stick. When the reaction wheel reaches a desired rpm, release the stick and test changing the rpm’s influence on the stick. Test when the stick is tilted can the motor change torque quickly enough to rebalance the stick.  When the assembly passes this test, it needs to be tested when implemented on the bike with wheels and other components installed. |
| R3. The Propelling motor is strong enough to power the bike, and the chain/belt is strong enough to hold the torque | V3. Implement the system onto the bike and test it with components and weights on the bike. If it can propel the bike on a 20-degree slope. |
| R4. The steering servo is quick and precise | V4. Test the lag of servo input and test the torque it can offer. |
| R5. The software logistic controls 2 11V motors properly based on tilted angle | V5. Test the subsystem before implementing the subsystem onto the bike model.  Test if the Motor Control System alters the inputs of two motors.  Test if the motor can withstand sudden changes when the reaction wheel is attached, and test if we can use the controller to control this subsystem. |
| R6. The motor can output a torque that is strong and quick enough to balance the bike model | V6. Implement the above design on our bike model with the battery and other two motors installed and test if the motor control system is quick and precise enough to correct tilts |

**Computing Subsystem**

|  |  |
| --- | --- |
| **Requirements** | **Verification** |
| R1. The microcontroller can interpret the correct tilt angle based on measurement. | V1. Set fixed angles like 0, 30, 60, and 90 degrees to see if the angle we get is the same as we expected. |
| R2. The noise of the readings is within acceptable range (+- 0.1 degrees) | V2. Test with several given angles and stabilize at that angle, then observe the angle changes among timestapes. |

**Power Subsystem**

|  |  |
| --- | --- |
| **Requirements** | **Verification** |
| R1. The 11.1V to 5V buck voltage regulator should output stable 5V | V1. Utilize a multimeter to gauge the input and output voltage and current, verifying they fall within predefined thresholds. |
| R2. The 5V to 3.3V linear voltage regulator should output stable 3.3V | V2. Utilize a multimeter to gauge the input and output voltage and current, verifying they fall within predefined thresholds. |
| R3. The 11.1V LiPo battery should supply the whole project >= 30min | V3. Use a timer to ensure it lasts for at least 30 minutes. Utilize a multimeter to gauge the input and output voltage and current, verifying they fall within predefined thresholds. |