MEMORANDUM

To: Prof. Charlie Refvem - crefvem@calpoly.edu
Prof. Eric Espinoza-Wade - erwade@calpoly.edu
Department of Mechanical Engineering, Cal Poly SLO

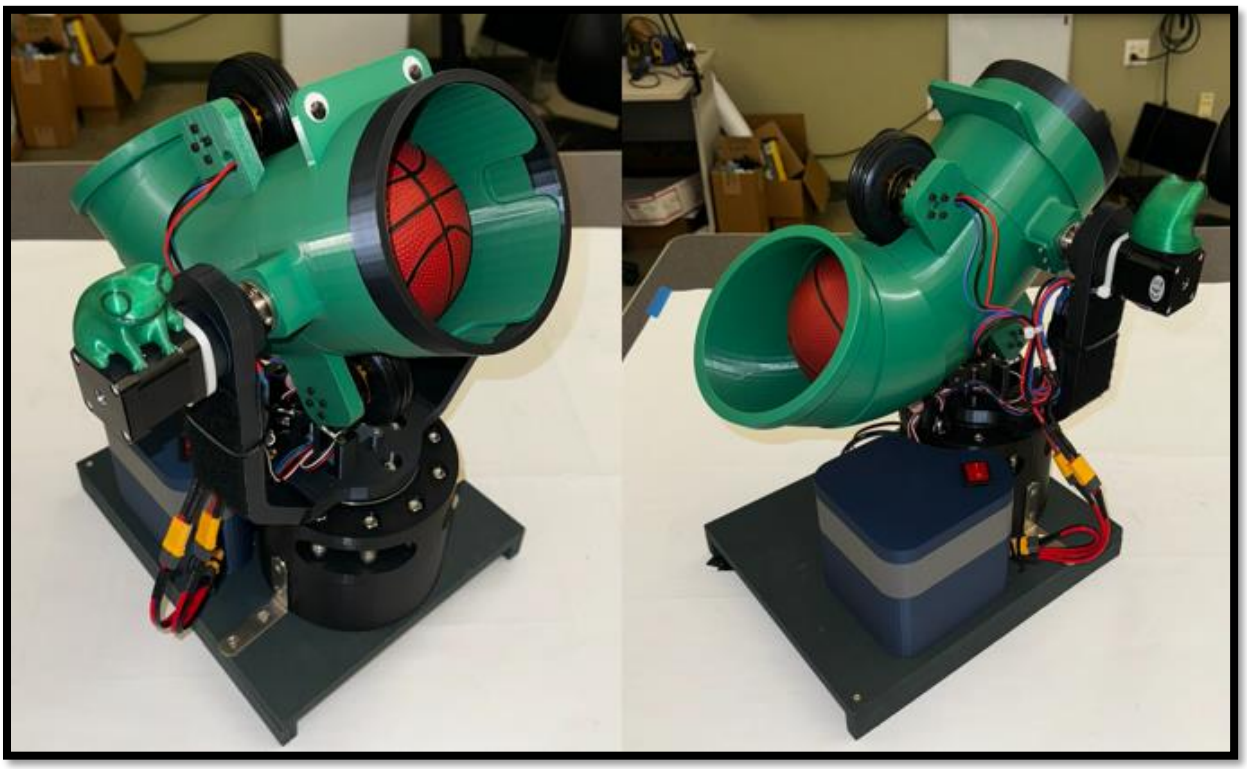
From: Jonathan Lam – jlam94@calpoly.edu
Vinh T. Vo – vvo11@calpoly.edu
Department of Mechanical Engineering, Cal Poly SLO

Course: ME 507 - Mechanical Control System Design

Group: MECHA08

Date: 05/10/2024

RE: Term Project Memo – Basketball Laucher



ABSTRACT

This report provides a detailed summary of the Basketball Launcher project from the first ideation stage to completion. After the project proposal, the team began by creating the mechanical design using SOLIDWORKS and successfully built the physical model within two weeks. Following this, the team focused on designing custom PCBs for both the launcher and its controller. Simultaneously, while the PCB design was underway, the team also developed the necessary software for both the controller and the launcher. This approach allowed for the software to be directly implemented onto the new PCB as soon as it arrived.

The purpose of this report is to document the entire project's process, including the mechanical design, electronic design, and programming efforts. It also highlights some of the challenges encountered during the project and how they were addressed. Additionally, the report includes most of the materials and documents related to the project, which are attached as appendices for further reference. This comprehensive overview aims to provide a clear understanding of the project's development

Table of Contents

ABSTRACT 1
OBJECTIVE 3
OVERVIEW 3
IDEATION 4
MECHANICAL DESIGN 5
CONTROLLER 6
LAUNCHER 7
CYCLOIDAL GEARBOX 7
LAUNCHER BODY 8
PCBA DESIGN 10
CONTROLLER 11
LAUNCHER 12
SOFTWARE IMPLEMENTATION 14
CONTROLLER 14
LAUNCHER 15
APPENDIX A – CONTROLLER PCB 16
APPENDIX B1 – LAUNCHER PCB – PAGE 1 17
APPENDIX B2 – LAUNCHER PCB – PAGE 2 18
APPENDIX B3 – LAUNCHER PCB – PAGE 3 19
APPENDIX B4 – LAUNCHER PCB – PAGE 4 20
APPENDIX C – CONTROLLER CODE IMPLEMENTATION 21
APPENDIX D – LAUNCHER CODE IMPLEMENTATION 22
APPENDIX E – MECHANICAL BILL OF MATERIAL 25
APPENDIX F – ELECTONIC BILL OF MATERIAL 27

OBJECTIVE

OVERVIEW

The Term Project in ME 507 requires students to design, build, program, test, and document an intelligent machine following specific guidelines. The project must include a custom PCB with an STM32F411 MCU (or similar), programmed in C, C++, or Rust. It should have two or more actuators, unique sensors, a closed-loop control algorithm, and a wireless controller for hands-free command or as a wireless emergency stop (e-stop).

The project also needs to fill up high construction quality and adheres to ME 507 lab safety rules, ensuring safety for all. Allowed manufacturing techniques include 3D printing, PCB fabrication, laser-cutting, and water-jetting for flat parts. Safety requirements include communicating with instructors before using or charging batteries and incorporating an emergency stop feature that activates communication with the wireless controller is lost.

IDEATION

The design is inspired by the Peashooter from the Plants vs. Zombies mobile game. The overall design of the two main assemblies in the project is sketched and listed below for reference. The sketch represents a rough idea of how all components are connected

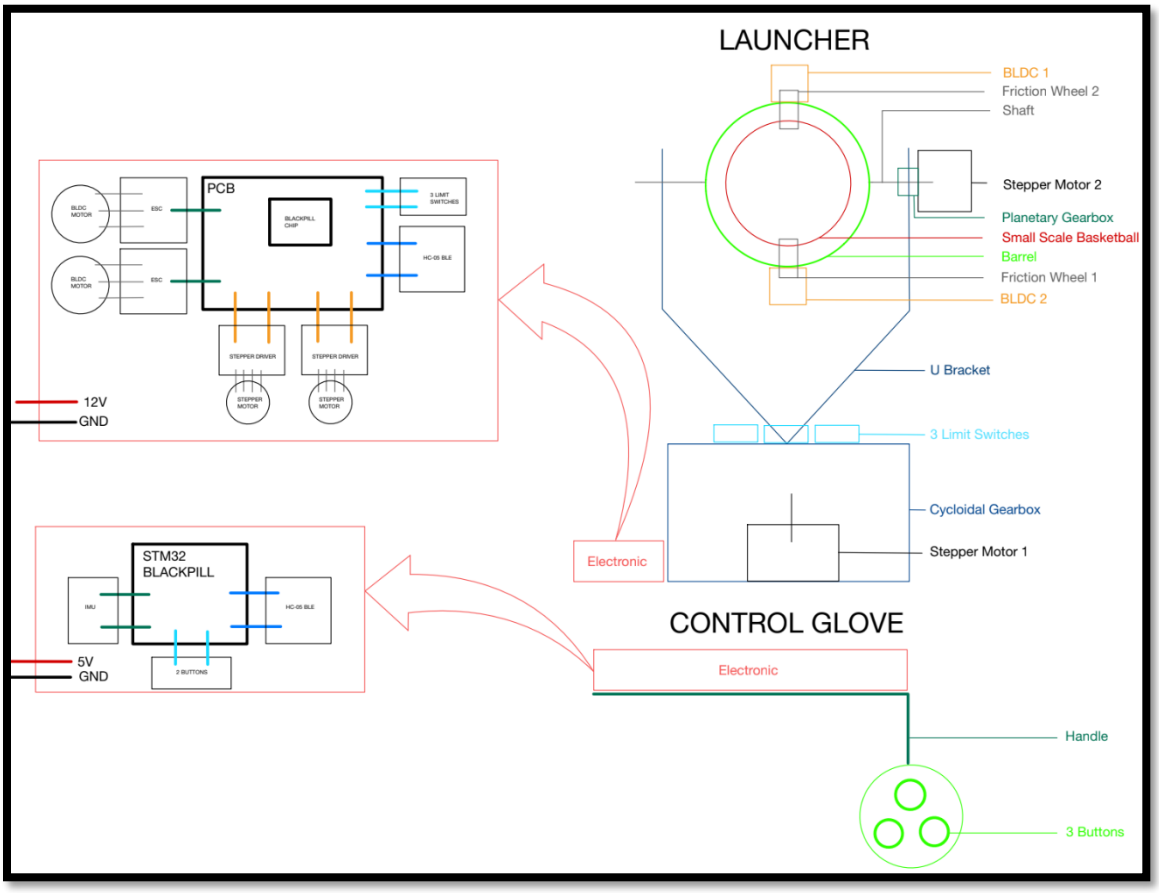


Figure 1. Ideation Sketch

The project is divided into two main assemblies:

- The controller will use an STM32 Black Pill microcontroller with an MPU6050 imu sensor attached to the user's hand. The controller will send signals via Bluetooth to the launcher, allowing the launcher to move according to the imu sensor and launch the ball.
- The launcher will be the main assembly of the project, it receives data from the controller and moves in 2 DOF (yaw & pitch). It will launch the basketball when the user pushes the button from the controller. A custom PCB centered around the STM32F401CEU6 chip is also a main requirement for this project and acts as the centerpiece of all mechanical components.

MECHANICAL DESIGN

The CAD modeling phase of the Basketball Launcher project was completed ahead of schedule, allowing the primary focus to shift to electronics and programming. Early completion of the CAD design ensured sufficient time to optimize the PCBA

The final CAD model, shown below, features the detailed design of the launcher with the controller conveniently attached to the side. Completing the CAD modeling early also helped identify and address potential mechanical issues, providing a solid implementation of electronic and programming tasks, thereby ensuring the project proceeded smoothly.

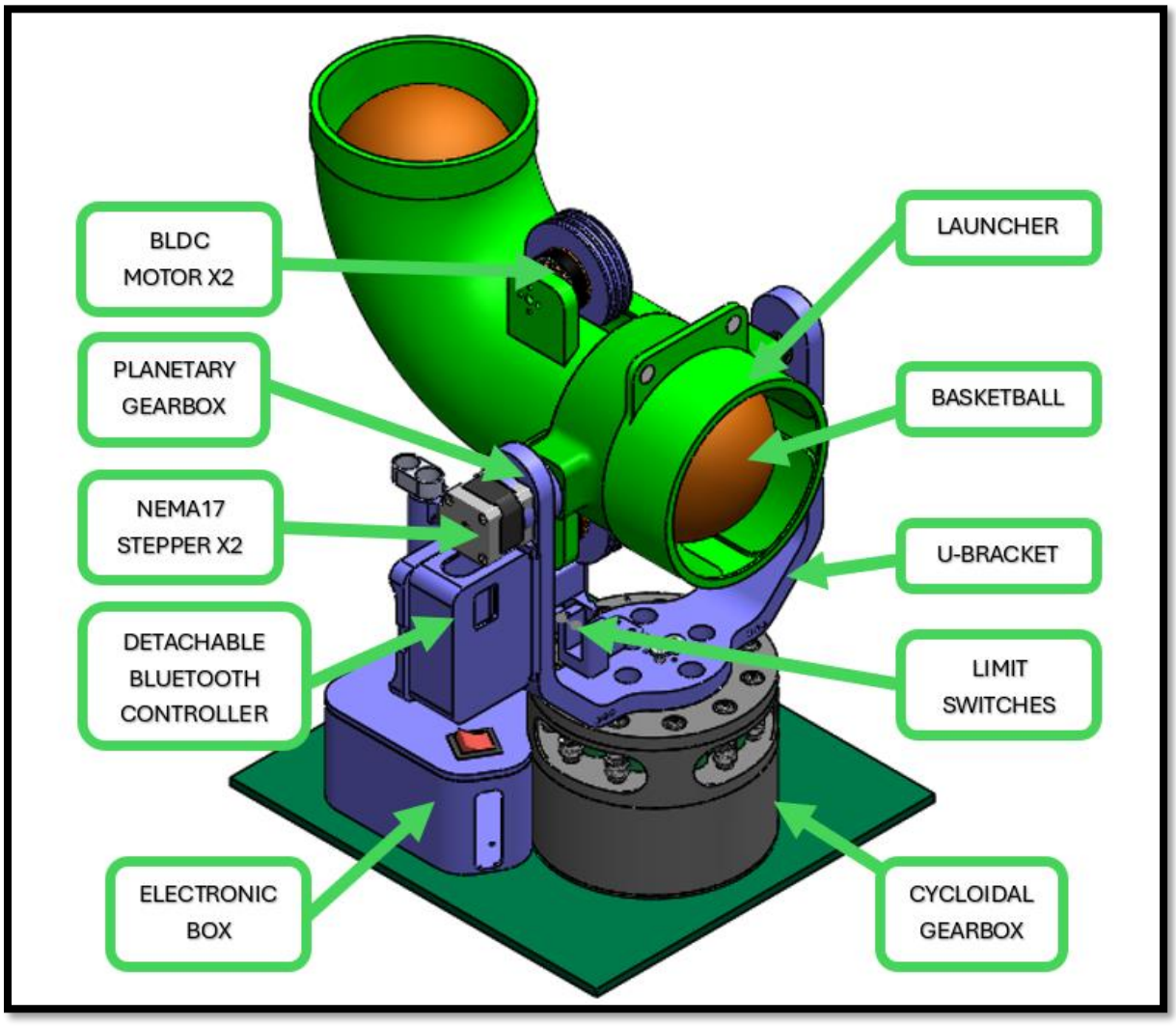


Figure 2. CAD Design

CONTROLLER

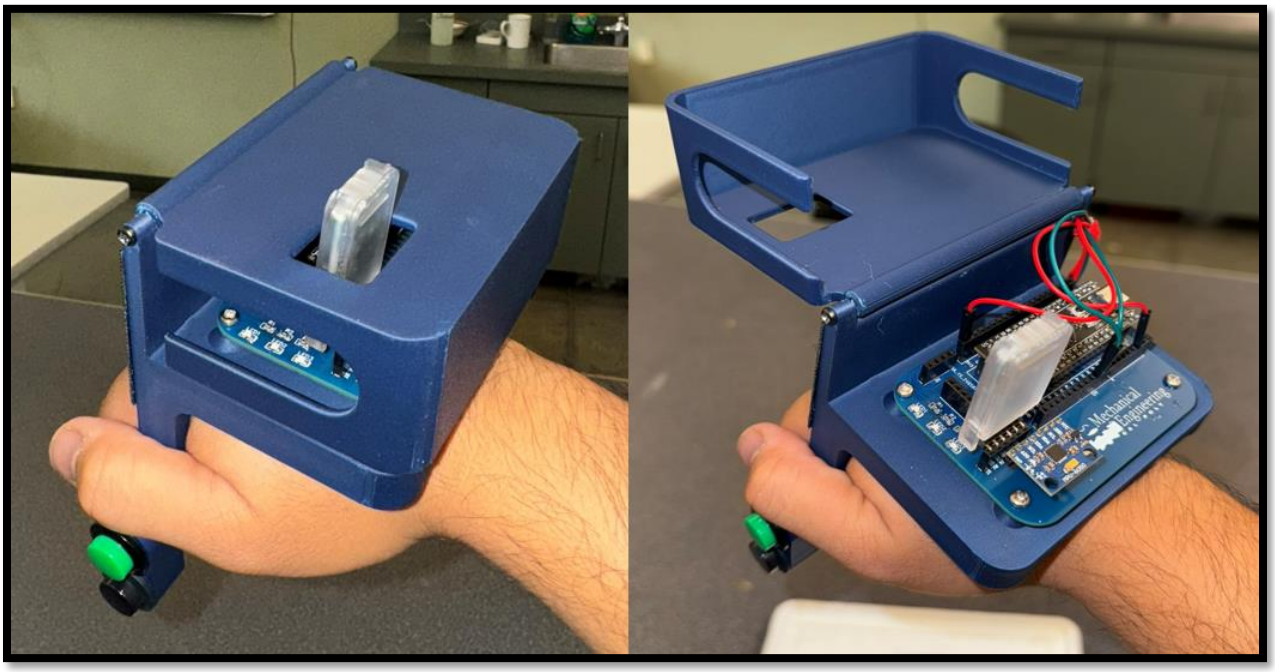


Figure 3. Controller

The controller housing includes two 3D-printed brackets connected by a hinge, allowing easy access to the PCBA inside. The controller features two buttons: a green button that puts the launcher into a moving mode, which follows the controller's motion via IMU data, and a black button that speeds up the BLDC motor, launching the ball. Note that the implemented code allows only one button to be pressed at a time; pressing both buttons simultaneously will result in no action unless user wants to trigger the launcher when it's in the Emergency Stop stage.

Additionally, there are three indicator LEDs for a user-friendly interface. One LED indicates the IMU connection status, while the other two LEDs indicate when either of the buttons is pressed. In terms of electronics and programming, the MPU6050 IMU uses the I2C communication protocol to send angular speed data to the STM32 BlackPill microcontroller. The microcontroller combines this data with the status of the two buttons and sends it to the HC-05 Bluetooth module in SLAVE mode via UART. The HC-05 then transmits this data to another HC-05 module in MASTER mode, which is connected to the launcher's custom PCBA.

More details about the code implementation will be discussed in the SOFTWARE IMPLEMENTATION section and Appendix C.

LAUNCHER
CYCLOIDAL GEARBOX

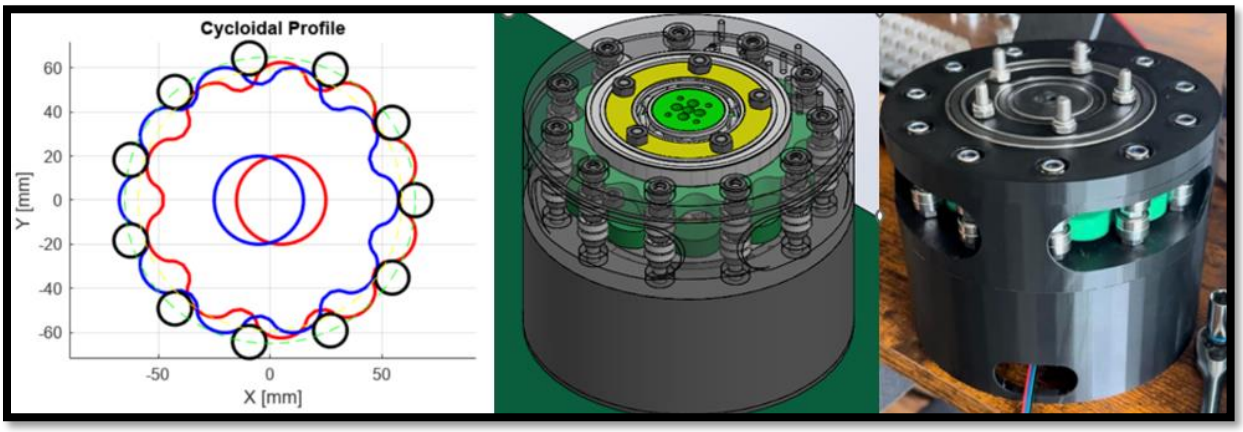


Figure 4. Cycloidal Gearbox

The concept of the cycloidal gearbox is inspired by the YouTube channel "How To Mechatronics." This gearbox aims to design a 3D-printed mechanism robust enough to support spinning the entire launcher's body, providing a 1:10 reduction gear ratio, and driven by a stepper motor.

The initial design was developed using a MATLAB script capable of calculating various bearing dimension scenarios. This aids in the sorting and purchasing of materials while minimizing costs.

The gearbox is constructed using M6 bolts that act as shafts for the 6mm inner diameter ball bearings of the cycloidal gearbox. Additionally, the assembly includes ball bearings with inner diameters of 25mm, 35mm, and 55mm, which are positioned on the eccentric center shaft.

The bottom housing is designed to integrate a NEMA 17 stepper motor, which inputs rotational motion. The output achieves a ten time increase in torque, which is then transmitted via five M6 bolts to the launcher's U-brackets.

LAUNCHER BODY

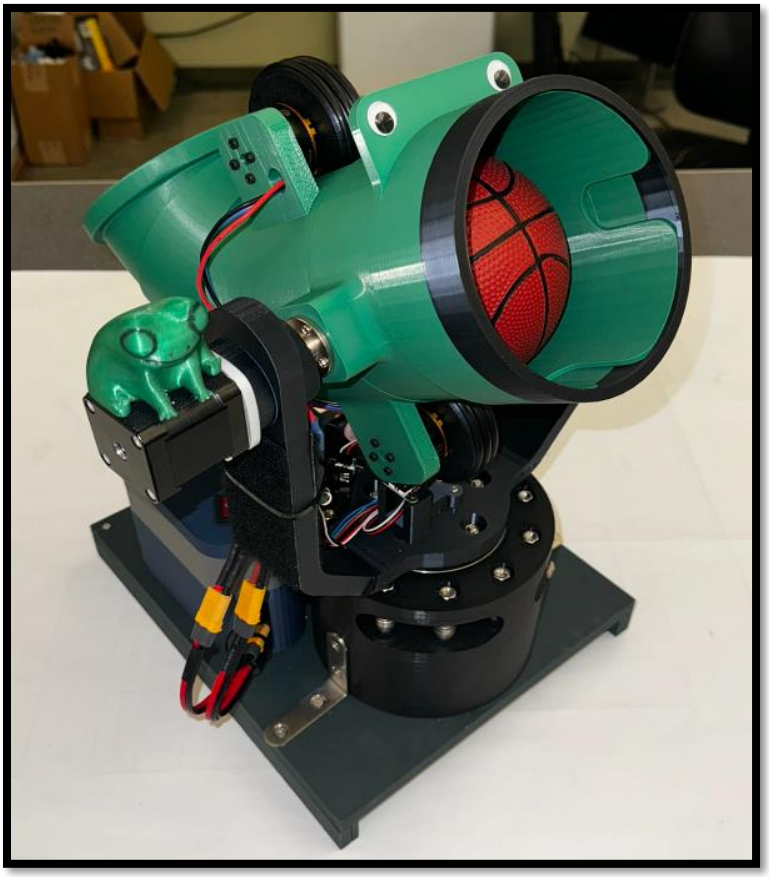


Figure 5. Launcher Body (Front View)

The launcher's body features a 3D-printed U-shaped bracket mounted on top of the cycloidal gearbox using five M6 bolts, which supports the launcher's 1st axis. Above the gearbox, a shaft and bearings subassembly provide support for the 2nd axis, which is also driven by a NEMA 17 stepper motor with an integrated 1:19 planetary gear reduction gearbox. This built-in gearbox ensures that the launcher maintains its static position when no motion is required, preventing it from tilting upwards due to the offset center of mass of the launcher in case the motor is disabled.

The launcher barrel is designed to launch two 5” small-scale basketballs, addressing both safety and focus requirements of the course. The basketballs are launched using two BLDC motors with a 650 KV rating, each equipped with a 3” diameter friction wheel with O-rings on the outside. These wheels rotate at 7215 rpm at 11.1VDC from a LiPo battery, theoretically providing a launch speed of approximately 95 ft/s under ideal, non-slip conditions. However, due to the safety concern, the BLDC motors’ speed is limited to 30% which corresponds to 28.5 ft/s in ideal, non-slip condition.

However, several factors can reduce the actual speed. The deformation of the ball absorbs some kinetic energy, friction along the barrel reduces speed, and aerodynamic drag affects the ball once it exits the barrel. The quality of the ESC driving the BLDC motors can also cause speed inconsistencies between 2 BLDC motors, further influencing the ball's trajectory due to drag and Magnus effect.

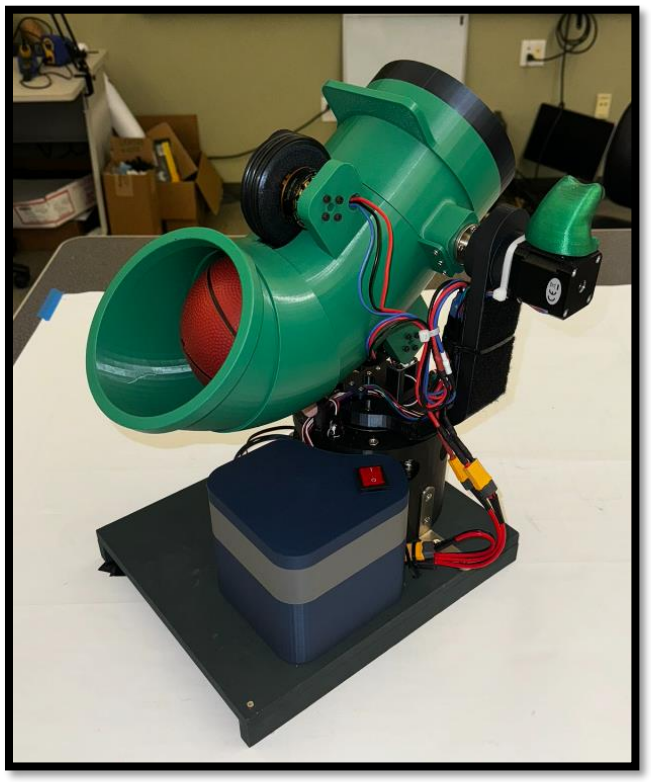


Figure 6. Launcher Body (Back View)

Below the barrel, there are three limit switches that restrict the motion of the launcher in both vertical and horizontal directions. These limit switches come with a built-in pull-up resistor and a B3B-XH connector, making it convenient to connect to the main PCBA. The electronic box houses the PCBA, an 11.1V LiPo battery encased in an explosion-proof LiPo battery safe bag, and all wire connections between components. The box is topped with a lid featuring a rocket switch to power the project. This electronic box not only shields the electronic components from external factors such as dirt and water but also protects the surroundings in case of a battery explosion.

PCBA DESIGN

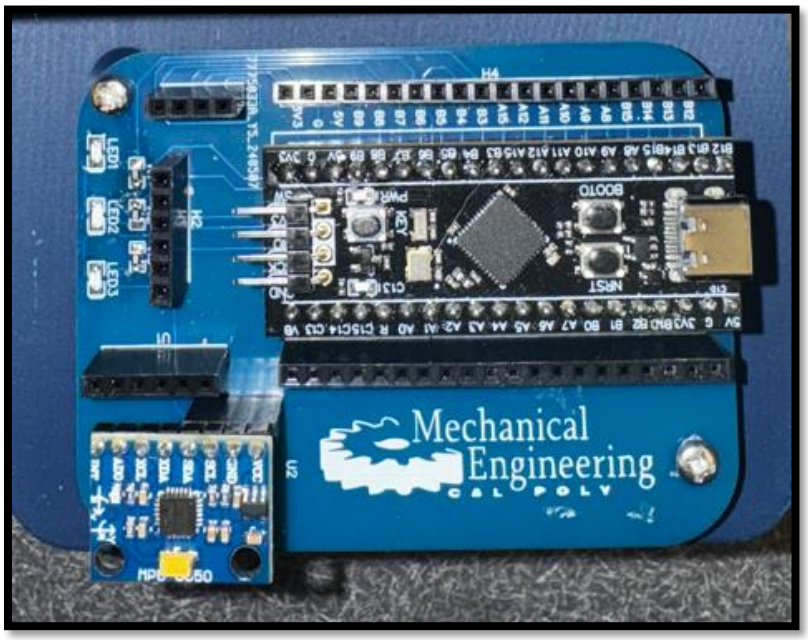


Figure 7. Controller PCB Assembly

Figure 8. Launcher PCB Assembly

CONTROLLER

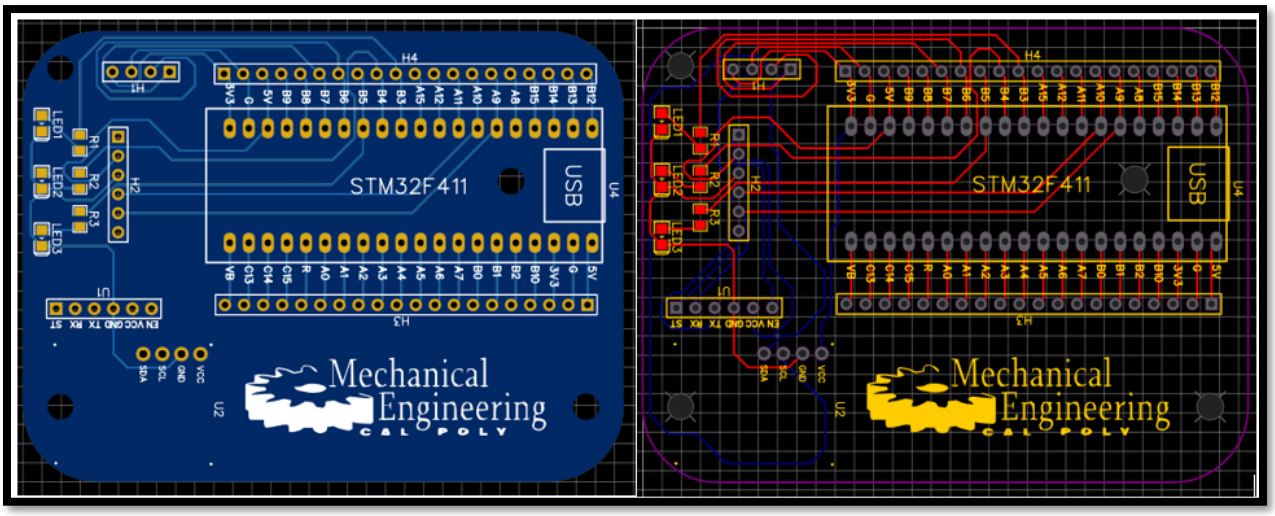


Figure 8. Controller PCB

In the controller's PCB, the connections are relatively simple compared to the launcher PCB. The controller's PCB primarily serves as an extended platform for mounting the STM32 BlackPill and the MPU6050, with internal wire trace connections for I2C communication to the BlackPill's I2C1 bus. The PCB also includes internal trace connections for the HC-05 Bluetooth module, connected to the BlackPill's UART1. Additionally, there are three surface-mounted indicator LEDs with resistors to display the IMU status and a button for user interaction.

The board is powered by a 5V USB-C cable, which can draw power from an external source such as a laptop, a 5V adapter, or a 5V battery. More details regarding the critical wire trace connections between the MCU and are described in the table below.

Table 1. Controller PCB Connectivity

MCU	Description	Connection
PA9	UART1 – TX	HC-05 – RX
PA10	UART2 – RX	HC-05 – TX
PB3	GPIO-B3	R1 – LED1 – GND
PB4	GPIO-B4	R2 – LED2 – GND
PB5	GPIO-B5	R3 – LED3 – GND
PB6	I2C1 – SCL	MPU-6050 - SCL
PB7	I2C1 – SDA	MPU-6050 - SDA

Note: For detailed PCB schematic, refer to Appendix A, which attached at the end of this report**.**

LAUNCHER

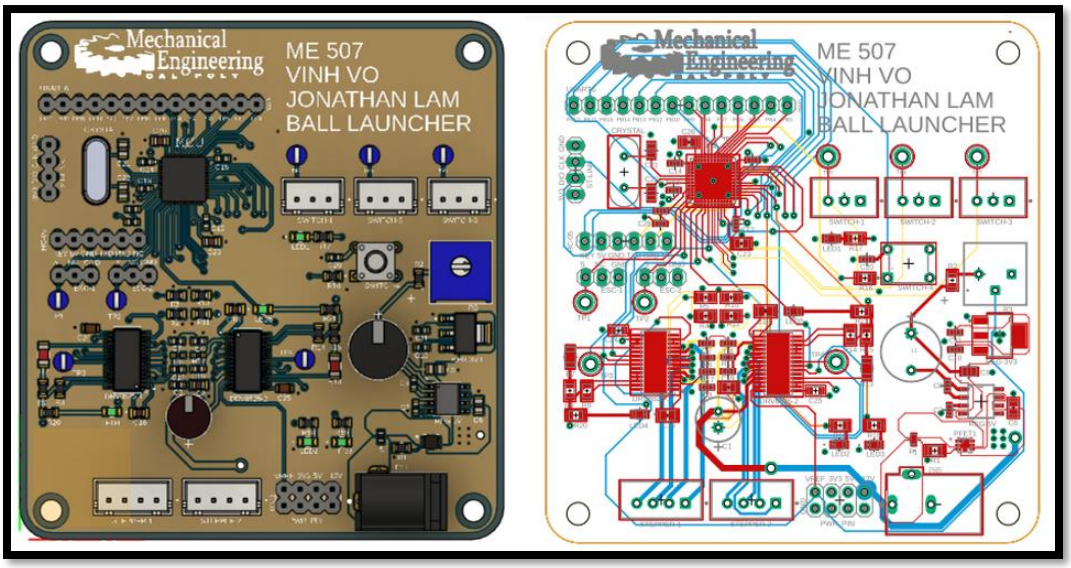


Figure 9. Launcher PCB

The launcher PCB is the project's centerpiece, consuming the most effort and time for optimization and revisions. Due to time constraints, students could only purchase the fabricated PCB once. Overall, the PCB including 4 separate area that is describe in Appendix B1 → B4 as follow:

- The 1st area (Appendix B1): located at the bottom right corner of the PCB is the power management section, which includes all components related to the power source. This area features a fuse and a PMOSFET for circuit protection in case the polarity of the 12 VDC power source is reversed. It also includes a switch regulator to step down from 12VDC to 5VDC, followed by a linear regulator to further reduce the voltage from 5VDC to 3.3VDC. Additionally, this section contains the necessary resistors, capacitors, and other components required for the operation of these regulators.

- The 2nd area (Appendix B2): located in the top left corner of the PCB is the section dedicated to the STM32F411CEU6 MCU chip. This area includes all necessary bypass capacitors for filtering the input power to the MCU, a typical reset circuit, and a 25 MHz crystal to support the MCU's functionality. Additionally, this section contains three indicator LEDs for user interface and required resistors, capacitors for the chip

- The 3rd area (Appendix B3): Located in the bottom left area of the PCB is the section focused on the two DRV8825 stepper motor driver chips, which control and run the two NEMA 17 stepper motors mentioned in the launcher description. These drivers are paired with potentiometers that can provide a voltage signal from 0-0.45VDC to the VREF pin, thereby limiting the current through the stepper motors from 0-0.9A. Additionally, this section includes all the necessary resistors and capacitors required for the operation of these two drivers.

- The 4th area (Appendix B4): this section also includes additional components such as connectors, test pads, and unused pins distributed across the PCB for potential use.

The board is powered by a 12 VDC power source, which can draw power from a 12 VDC adapter or an 11.1 V Lipo battery. More details regarding the critical wire trace connections between the MCU and are described in the table below.

Note: For detail PCB schematic, refer to Appendix B1 → B4, which attached at the end of this report**.**

Table 2. Launcher PCB Connectivity

MCU	Description	Connection
PA1	PWM TIM2_CH2	ESC-1
PA2	PWM TIM2_CH2	ESC-2
PA3	GPIO OUTPUT	EN_1 (DRV8825)
PA4	GPIO OUTPUT	EN_2 (DRV8825)
PA5	GPIO OUTPUT	DIR_1 (DRV8825)
PA6	GPIO OUTPUT	DIR_2 (DRV8825)
PA7	GPIO OUTPUT	STEP_1 (DRV8825)
PA8	GPIO OUTPUT	STEP_2 (DRV8825)
PA9	USART1_TX	HC-05 – RX
PA10	USART1_RX	HC-05 – TX
PA11	USART6_TX	RX (DEBUG)
PA12	USART6_RX	TX (DEBUG)
PA13	SWDIO	SWDIO (ST-LINK)
PA14	SWCLK	SWCLK (ST-LINK)
PH0	OSC-IN	OSC-IN (CRYSTAL)
PH1	OSC-OUT	OSC-OUT (CRYSTAL)
PB0	GPIO OUTPUT	Switch 1
PB1	GPIO OUTPUT	Switch 2
PB2	GPIO OUTPUT	Switch 3

NOTE:

Since the team only have 1 chance to order the PCBA for the launcher, there are several modifications are made so that the PCBA can operate appropriately:

- Remove C27 and C28, then short those pins so that VREF+ pin → 3V3 net.
- Replace R2 from 50kΩ to 10kΩ to reduce the impedance of VREF signal line.

Additionally, the connectors of the two stepper motors need to be adjusted to ensure the cable order is consistent with PCBA connectors, as shown in the figure below

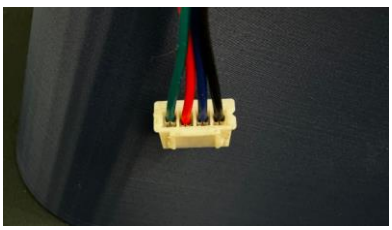


Figure 10. Cable Configure

SOFTWARE IMPLEMENTATION

CONTROLLER

Table 3. Controller Finite State Machine

State	Description	Actions	Transitions
STATE0	INIT	- Attempt to connect - Turn on LED1 if successful	- If unsuccessful, go → STATE1 - If successful, go → STATE2
STATE1	ERROR	- Toggle LED1 if failed to connect to the IMU	- Always go → STATE0
STATE2	IMU	- Continuously read the data from the IMU	- Always go → STATE3
STATE3	BUTTON_LED	- Read the buttons + update the LEDs	- Always go → STATE4
STATE4	TRANSFER	- Send data to UART1 & UART6 (debug)	- Always go → STATE2

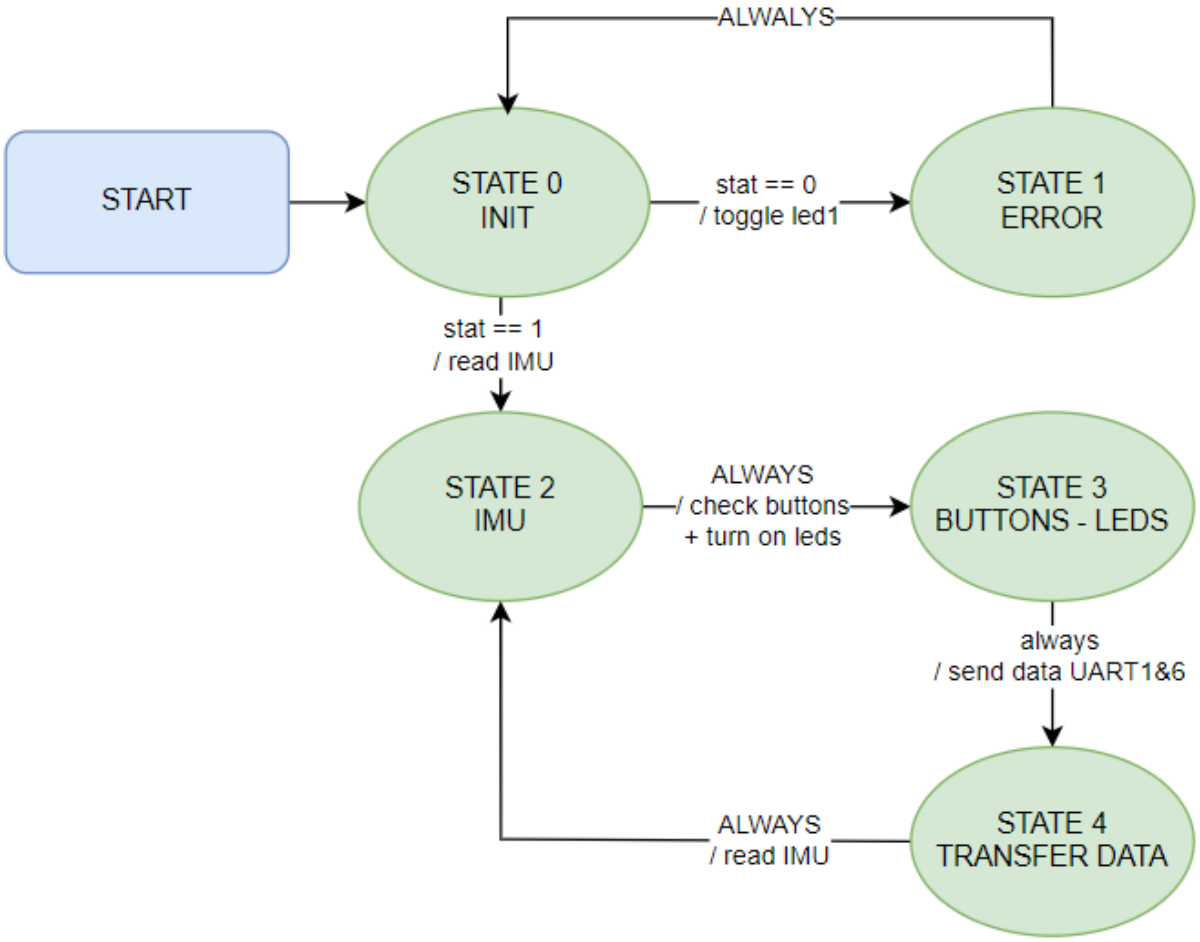


Figure 11. Controller Finite State Machine

LAUNCHER

Table 4. Launcher Finite State Machine

State	Description	Actions	Transitions
STATE0	INIT	Init all components	Always go → STATE0
STATE1	DECISION HUB	Check condition and branch to different states	If S1 = 1 || S2 = 1 || S3 = 1→ STATE3 ESTOP = 1 → STATE2 MOVE = 1 & SHOT = 0 → STATE3 MOVE = 0 & SHOT = 1 → STATE4
STATE2	ESTOP	Stop all components	If dT < 5000 → STATE2 dT > 5000 → STATE1
STATE3	STEPPER RUN	Step 2 steppers in different speed	Always go → STATE1
STATE4	BLDC RUN	Spin 2 BLDC Motor at 30% speed	Always go → STATE1

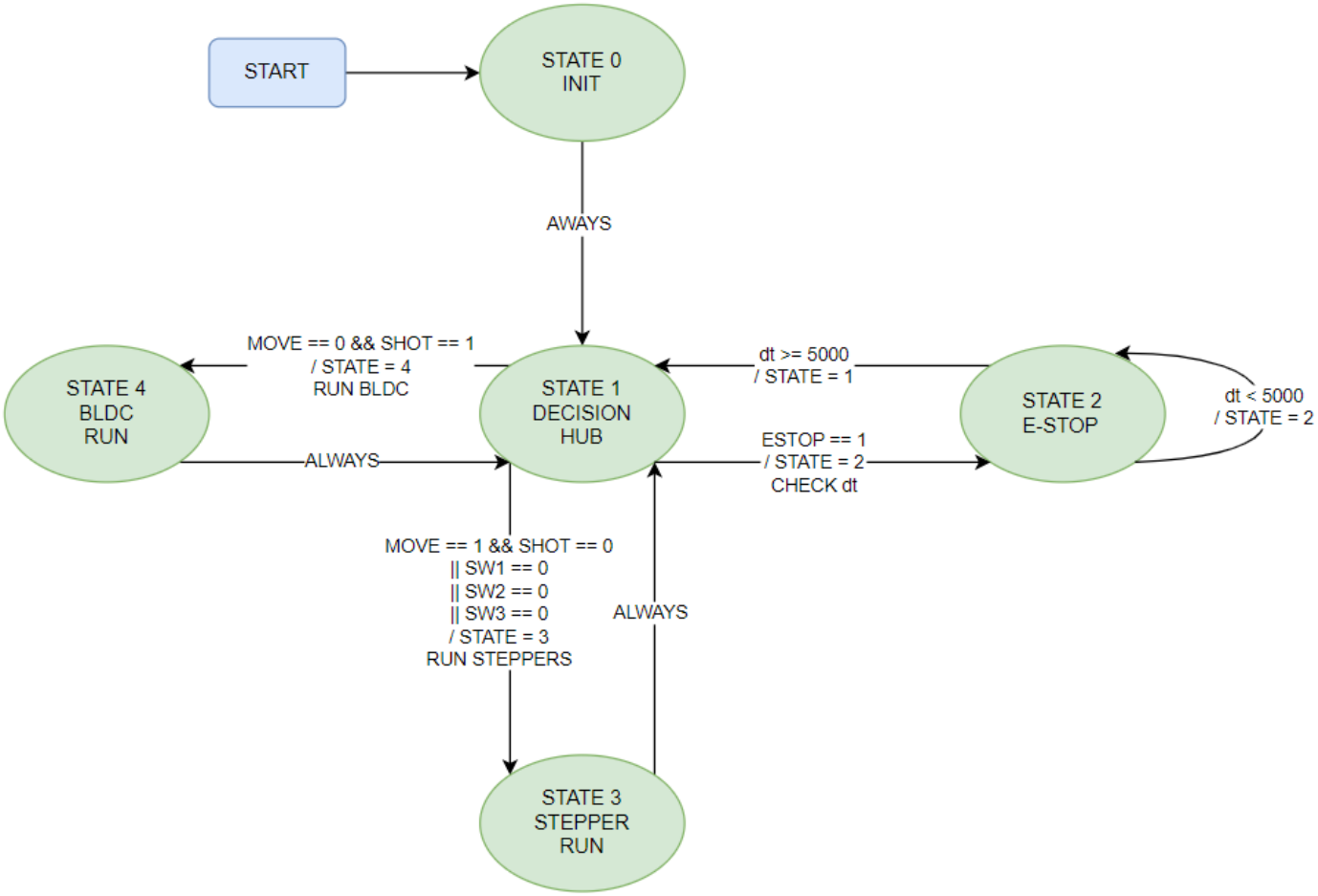
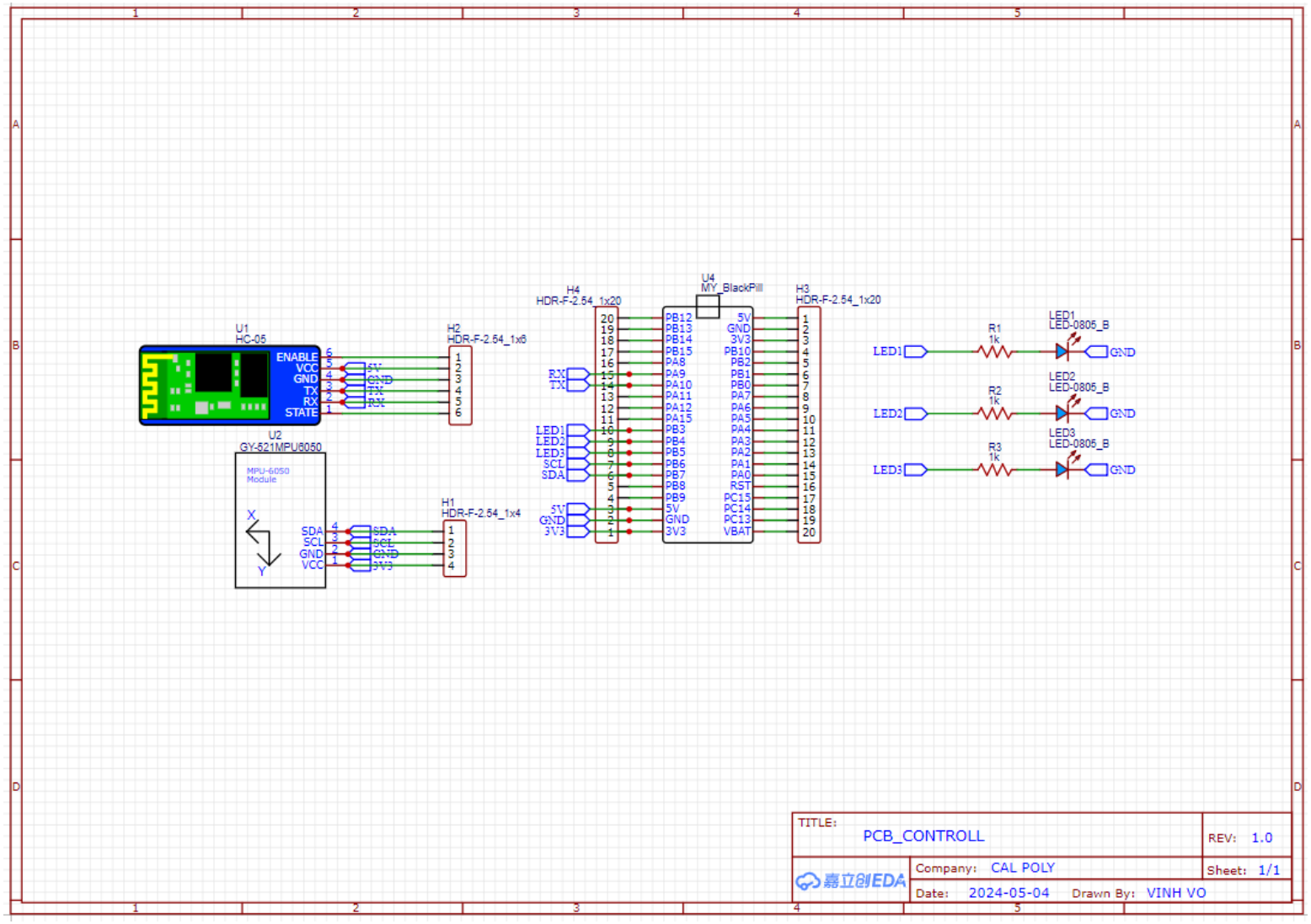
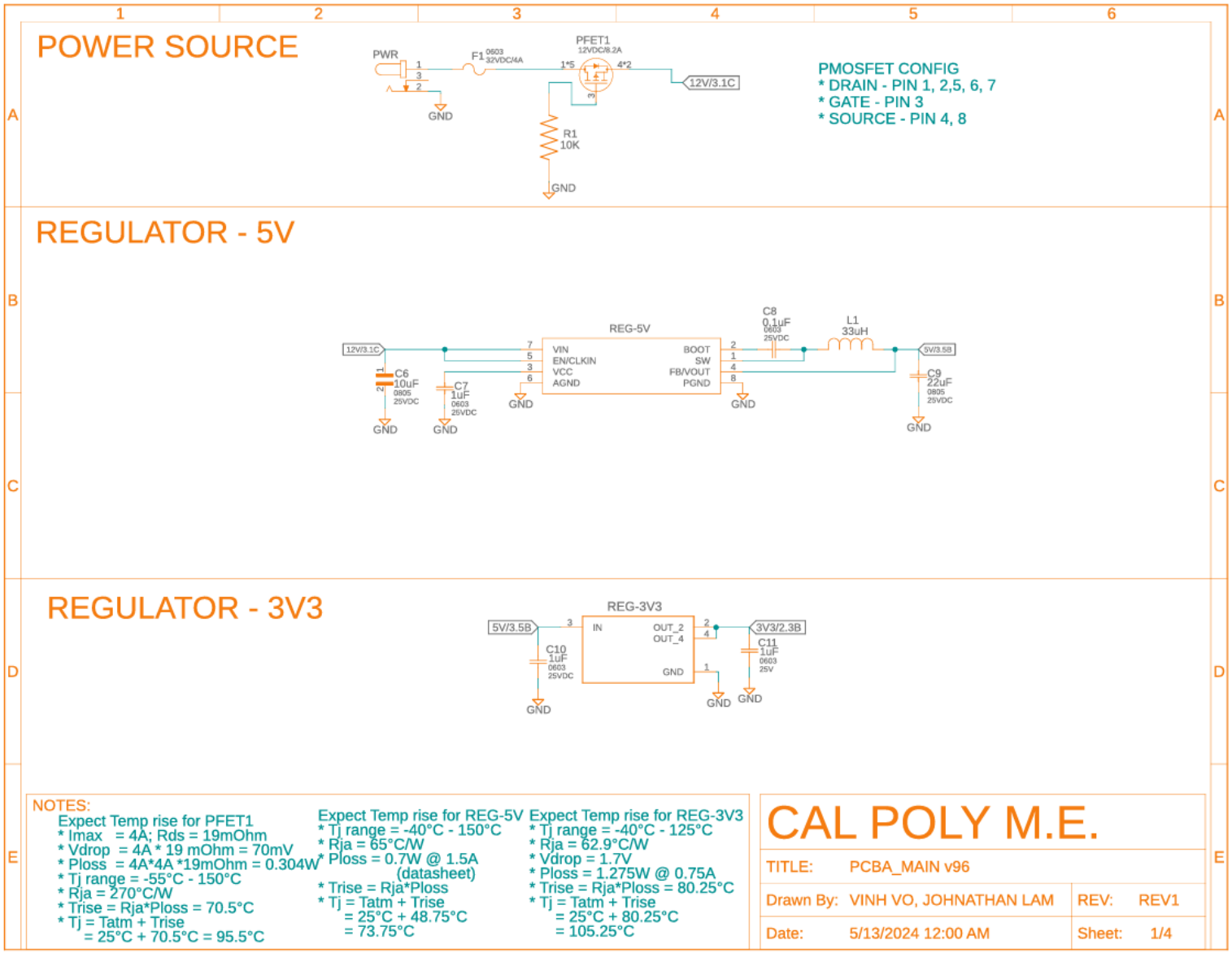


Figure 12. Controller Finite State Machine

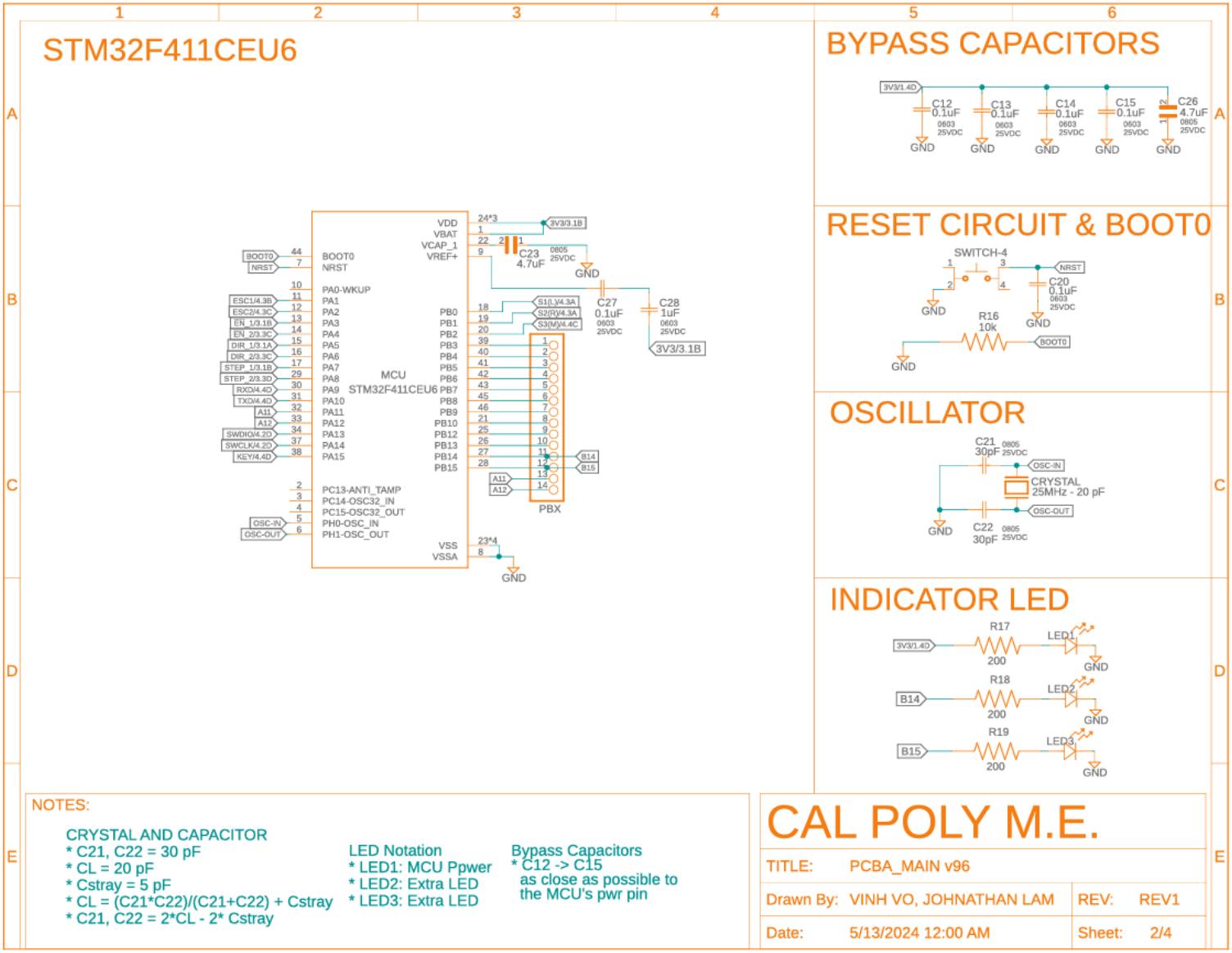
APPENDIX A – CONTROLLER PCB



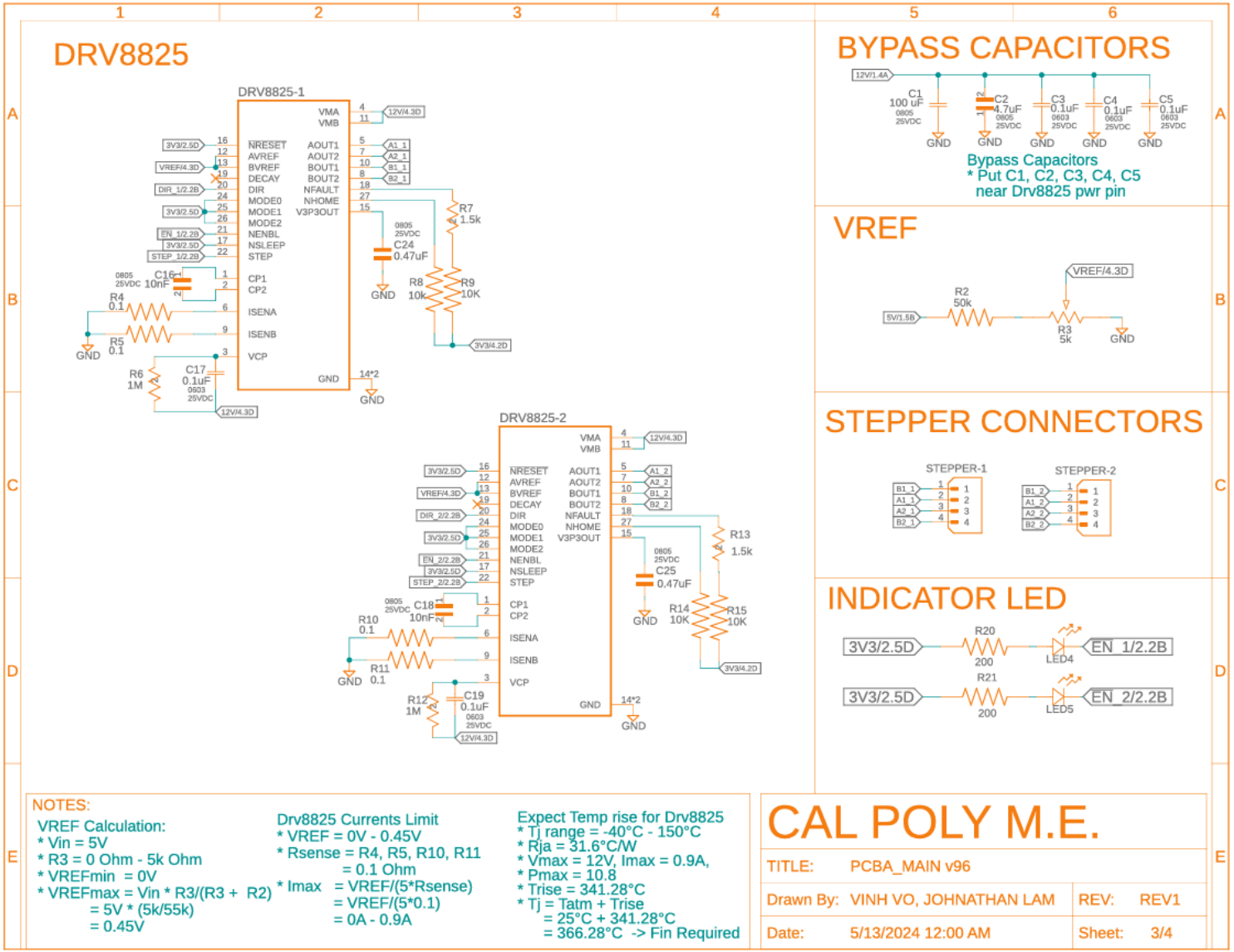
APPENDIX B1 – LAUNCHER PCB – PAGE 1



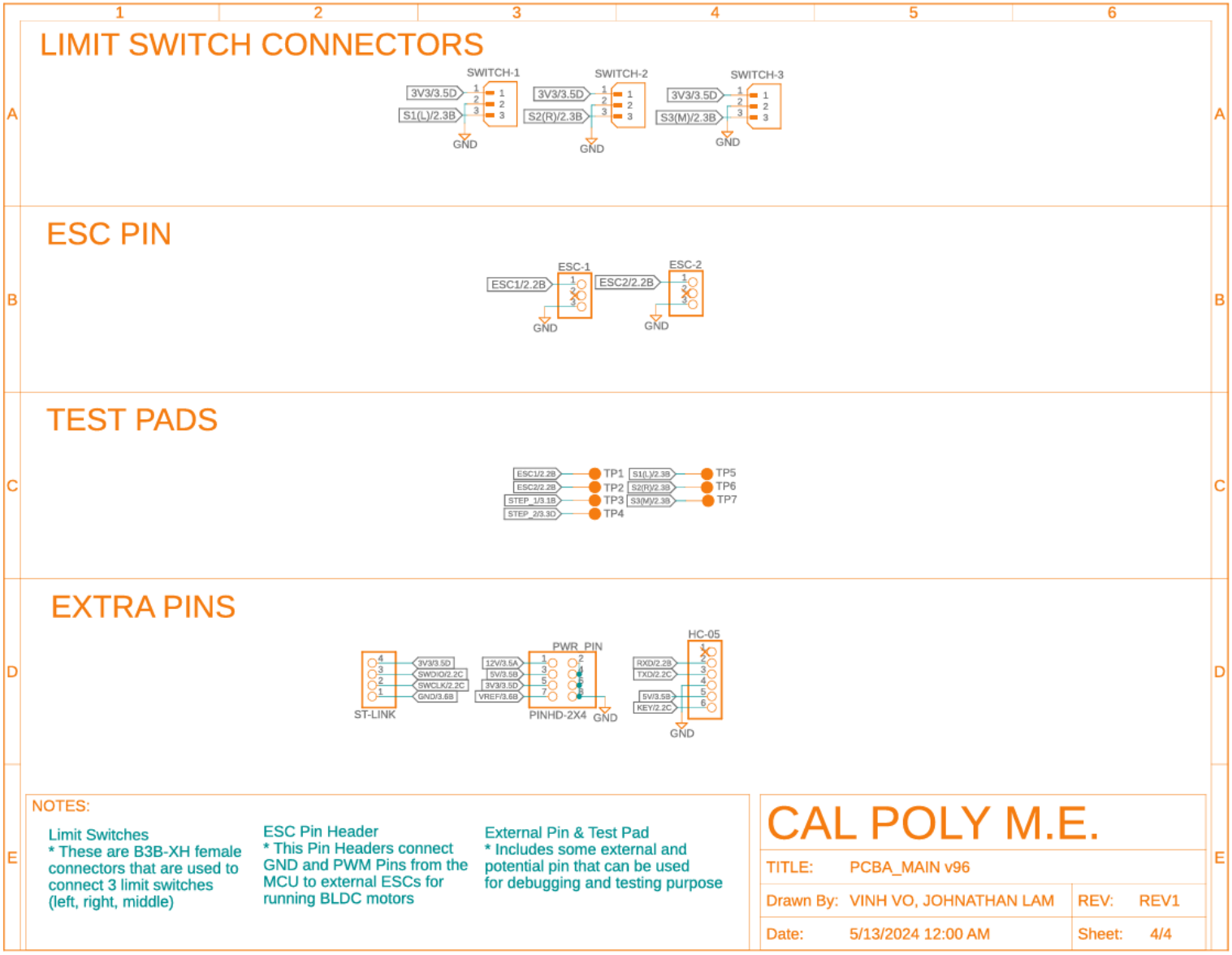
APPENDIX B2 – LAUNCHER PCB – PAGE 2



APPENDIX B3 – LAUNCHER PCB – PAGE 3



APPENDIX B4 – LAUNCHER PCB – PAGE 4



APPENDIX C – CONTROLLER CODE IMPLEMENTATION

```
while (1) {
    switch (STATE) {
            case 0: // STATE_INIT
                if (stat) {
                  stat = mpu6050_init(&imu, &hi2c1);
                  HAL_Delay(500);
                  mpu6050_calibrate(&imu);
                  HAL_Delay(500);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, stat);
                  STATE = STATE_2_IMU;
                } else {
                    STATE = STATE_1_ERROR;
                }
                break;
            case 1:
                HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_3);
                HAL_Delay(200);
                STATE = STATE_0_INIT;
                break;
            case 2:
                mpu6050_update(&imu);
                gX = mpu6050_get_gX(&imu);
                gY = mpu6050_get_gY(&imu);
                gZ = mpu6050_get_gZ(&imu);
                STATE = STATE_3_BUTTON_LED;
                break;
            case 3:
                shot = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_4);
                move = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_5);
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, shot); // LED2
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, move); // LED3
                STATE = STATE_4_TRANSFER;
                break;
            case 4:
                snprintf(buffer, sizeof(buffer), "%d\t%d\t%d\t%d\r\n", (int)move, (int)shot,
gZ, gY);
                HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), 100);
                HAL_UART_Transmit(&huart2, (uint8_t *)buffer, strlen(buffer), 100);
                HAL_Delay(10);
                STATE = STATE_2_IMU;
                break;
        }
}
```

APPENDIX D – LAUNCHER CODE IMPLEMENTATION

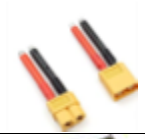
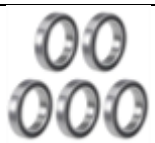
```
while (1)
  {
      // RUN FSM
      // STATE 0: INIT ALL OBJECTS
        if (STATE == STATE_0_INIT) {
              // INIT RADIO TIMER
              HAL_TIM_Base_Start(&htim4);
              HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1);
              HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_2);
              // INIT STEPPER MOTOR
              Stepper_init(&STEPPER1, GPIOA, GPIO_PIN_3, GPIO_PIN_5,
GPIO_PIN_7);
              Stepper_init(&STEPPER2, GPIOA, GPIO_PIN_4, GPIO_PIN_6,
GPIO_PIN_8);
              // INIT D4215 BLDC MOTORS
              D4215_init  (&ESC1,&htim2,TIM_CHANNEL_2);
              D4215_init  (&ESC2,&htim2,TIM_CHANNEL_3);
              // INIT 3 LIMIT SWITCHES
              Switch_init (&S1, GPIOB, GPIO_PIN_0);
              Switch_init (&S2, GPIOB, GPIO_PIN_1);
              Switch_init (&S3, GPIOB, GPIO_PIN_2);
              // INIT 2 EXTRA LEDS
              LED_init    (&LED1, GPIOB, GPIO_PIN_14);
              LED_init    (&LED2, GPIOB, GPIO_PIN_15);
              // MOVE --> STATE 1: DECISION HUB
              STATE = STATE_1_HUB;
        }
        // STATE 1: DECISION HUB TO BRANCH TO DIFFERENT HUB
        else if (STATE == STATE_1_HUB) {
              // ENABLE 2 STEPPERS
              Stepper_enable(&STEPPER1);
              Stepper_enable(&STEPPER2);
              // READ 3 LIMIT SWITCHES
              SW1   = Switch_getStatus(&S1);
              SW2   = Switch_getStatus(&S2);
              SW3   = Switch_getStatus(&S3);
              // CHECK ESTOP
              if (ESTOP == 1) {
                    ti = HAL_GetTick();
                    STATE = STATE_2_ESTOP;
              }
```

```
            // CHECK LIMIT SWITCHES OR "MOVE" FLAG FROM CONTROLLER
            else if ((MOVE == 1 && SHOT == 0) || SW1 == 0 || SW2 == 0 ||
SW3 == 0) {
                  if (SW1 == 0) {
                      SW = 1;
                      angleTarget = 10;
                      yawDirectionSW = 0;
                  } else if (SW2 == 0) {
                      SW = 2;
                      angleTarget = 10;
                      yawDirectionSW = 1;
                  } else if (SW3 == 0) {
                        SW = 3;
                      angleTarget = 10;
                      pitchDirectionSW = 0;
                  } else {
                        angleTarget = 10;
                        SW = 0;
                  }
                  STATE = STATE_3_STEPPER;
            }
            // CHECK "SHOT" FLAG FROM CONTROLLER
            else if (MOVE == 0 && SHOT == 1) {
                  STATE = STATE_4_BLDC;
            }
            // STOP IF
            else {
                  D4215_set(&ESC1, 0);
                  D4215_set(&ESC2, 0);
                  STATE = STATE_1_HUB;
            }
      }
      // STATE 2: EMERGENCY STOP STATE
      else if (STATE == STATE_2_ESTOP) {
            // DISABLE STEPPER MOTORS
            Stepper_disable(&STEPPER1);
            Stepper_disable(&STEPPER2);
            // STOP BLDC MOTORS
            D4215_set(&ESC1, 0);
            D4215_set(&ESC2, 0);
            // CHECK FOR RESET BY HOLDING BUTTONS FOR 5 SECOND
            if (MOVE == 1 && SHOT == 1) {
                  tf = HAL_GetTick();
                  dt += (tf-ti);
                  ti = tf;
            }
            else {
                  dt = 0;
            }
            STATE = (dt > 5000) ? STATE_1_HUB : STATE_2_ESTOP;

      }
```

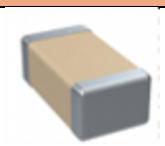
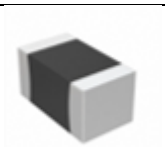
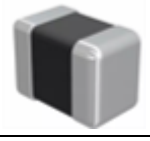
```
        // STATE 3: SPIN STEPPER MOTORS
        else if (STATE == STATE_3_STEPPER) {
              // LEFT SWITCH HIT --> MOVE TO THE RIGHT
              if (SW == 1) {
                    for (int i = 0; i <= angleTarget/2; i++) {
                          Stepper_setspeed (&STEPPER1, 10, yawDirectionSW);
                    }
                    SW = 0;
              }
              // RIGHT SWITCH HIT --> MOVE TO THE LEFT
              else if (SW == 2) {
                    for (int i = 0; i <= angleTarget/2; i ++) {
                          Stepper_setspeed (&STEPPER1, 10, yawDirectionSW);
                    }
                    SW = 0;
              }
              // MID SWITCH HIT --> MOVE UP
              else if (SW == 3) {
                    for (int i = 0; i <= 10 * angleTarget; i ++) {
                          Stepper_setspeed (&STEPPER2, 10, pitchDirectionSW);
                    }
                    SW = 0;
              }
              // IF NOT SWITCH, MOVE ACCODING TO THE IMU DATA
              else {
                    uint16_t yaw_map = map(yaw, 20, 250, 5, 30);
                    if (yaw > 20) {
                          for (int i = 0; i <= yaw_map; i ++) {
                                Stepper_setspeed (&STEPPER1,1,yawDirection);
                          }
                    }
                    uint16_t pitch_map = map(pitch, 20, 250, 10, 50);
                    if (pitch > 20) {
                          for (int i = 0; i <= pitch_map; i ++) {
                                Stepper_setspeed(&STEPPER2,1,pitchDirection);
                          }
                    }
              }
              STATE = STATE_1_HUB;
        }
        // STATE 4: BLDC MOTOR
        else if (STATE == STATE_4_BLDC) {
              D4215_set(&ESC1, 30);
              D4215_set(&ESC2, 30);
              STATE = STATE_1_HUB;
        }
        // STATE 1: DECISION HUB IN CASE SOME ERROR
        else {
              STATE = STATE_1_HUB;
        }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

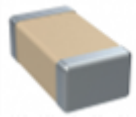
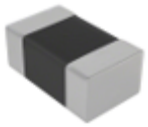
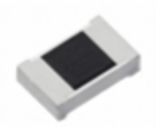
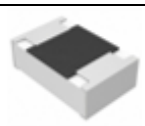
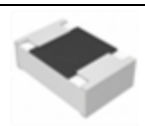
APPENDIX E – MECHANICAL BILL OF MATERIAL

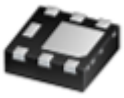
No	Picture	Parts	Unit Price	Qty	Total Amount
1		NEMA17	$ 15.00	1	$ 15.00
2		NEMA17 + GEAR BOX	$ 40.00	1	$ 40.00
3		D421 BLDC	$ 27.00	2	$ 54.00
4		UBEC ESC	$ 22.00	2	$ 44.00
5		O RING	$ 2.00	2	$ 4.00
6		SWITCH	$ 10.00	1	$ 10.00
7		BASKETBALL	$ 6.00	1	$ 6.00
8		HC-05	$ 10.00	2	$ 20.00
9		LIMIT SWITCH	$ 1.00	3	$ 3.00
10		CONNECTOR	$ 10.00	1	$ 10.00

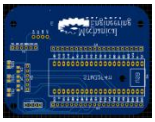
11		CONNECTOR	$ 3.00	2	$ 6.00
12		BATTERY	$ 18.00	2	$ 36.00
13		FILLAMENT	$ 20.00	4	$ 80.00
14		BEARINGS	$ 10.00	2	$ 20.00
15		BEARINGS	$ 10.00	3	$ 30.00
16		BEARINGS	$ 6.00	2	$ 12.00
17		BEARINGS	$ 10.00	1	$ 10.00
18		NUTS	$ 6.00	1	$ 6.00
19		NUTS	$ 10.00	1	$ 10.00
20		BOLTS	$ 10.00	2	$ 20.00

APPENDIX F – ELECTONIC BILL OF MATERIAL

Picture	Number Part	Buy	Cost
CONNECTOR			
	PJ-102AH	1	$ 0.70
	B3B-XH-A	4	$ 0.84
	B4B-XH-A	3	$ 0.69
CAPACITORS			
	C0805C103J1GEC7800	4	$ 0.58
	CC0603KRX7R8BB104	20	$ 2.00
	C2012X5R1E475K125AB	5	$ 0.95
	CC0805KRX5R8BB106	3	$ 1.14
	8.6002E+11	2	$ 0.26

	CC0603KRX5R8BB105	10	$ 1.00
	CC0805MKX5R8BB226	3	$ 1.23
	C0805C300M3GAC7800	5	$ 1.20
	CGA4J2X7R1E474K125AA	5	$ 1.20
RESISTORS			
	3386P-1-502LF	1	$ 1.77
	MCU08050D5002BP100	2	$ 1.70
	RC0805FR-0710KL	12	$ 1.20
	RC1206FR-071K5L	5	$ 0.50
	ERJ-3EKF1004V	5	$ 0.50
	ERJ-U6SFR10V	8	$ 5.28
	RC0805FR-07200RL	10	$ 1.00

OTHERS			
	MFU0603FF04000P500	2	$ 0.48
	RLB0914-330KL	2	$ 0.84
	TL1105BF160Q	2	$ 0.40
	631-FC4STCBMF25.0-ND	2	$ 0.68
	36-5127-ND	10	$ 3.80
	VAOL-S8SB4	6	$ 4.32
PCB + MCU + REGULATORS + MOTOR DRIVER			
	TLV1117LV33DCYR	2	$ 0.70
	PMPB15XP,115	2	$ 0.94
	L6981N50DR	1	$ 2.87
	STM32F411CEU6	1	$ 7.40

	DRV8825	2	$ 11.30
	LAUNCHER PCB	1	$ 72.28
	CONTROL PCB	1	$ 26.42