IRDH International Journal of Technology, Agriculture & Natural Sciences
https://irdhjournals.com/ijtans
Vol 2, No 1 (2025) : March. E-ISSN : 3032-2286

Integrating ESP32 and VESC-Controlled BLDC Motors in Cart Inverted Pendulum Design

Ida Lailatul Fitria[1], Indrazno Siradjuddin[2], Ferdian Ronilaya[3], Gillang Al Azhar[4], Zakiyah Amalia[5], Budhy Setiawan[6]
** Correspondence Author: idalailatul25@gmail.com*
[1,2,4,5]Electronics Engineering, State Polytechnic of Malang (Polinema), Malang, Indonesia
[3]Electrical Systems Engineering, State Polytechnic of Malang (Polinema), Malang, Indonesia
[6]Mechanical Engineering, State Polytechnic of Malang (Polinema), Malang, Indonesia

INDEXING

Keywords:
Keyword 1; Cart Inverted Pendulum;
Keyword 2; BLDC;
Keyword 3; Integrating ESP32;
Keyword 4; VESC Controlled;
Keyword 5; FreeRTOS;

ABSTRACT

This paper presents the design and implementation of a cart inverted pendulum system using an ESP32 microcontroller and a brushless DC (BLDC) motor with a Vedder Electronic Speed Controller (VESC). The cart inverted pendulum is a well-known benchmark problem in control theory due to its nonlinear, unstable, and underactuated nature. The ESP32 System-on-Chip (SoC) is chosen for its high processing power, low-cost, and low-power characteristics, making it suitable for real-time control applications. The BLDC motor, coupled with the VESC, provides efficient and precise actuation for the cart inverted pendulum system. The VESC, an open-source motor controller, offers advanced control algorithms and hardware upgrades that enable accurate current, position, and speed control. Integrating the ESP32 and BLDC motor with VESC in the cart inverted pendulum design addresses the need for more efficient and cost-effective solutions in this field. The paper discusses the hardware design, software design using FreeRTOS and serial communication, and presents the results of the hardware and software evaluation. The findings of this study contribute to the advancement of control theory and its applications in various domains, such as robotics and electric vehicles.

Article History
Received 18 November 2024; Revised 05 December 2024; Accepted 30 December 2024 ; Publish 01 March 2025

INTRODUCTION

The cart inverted pendulum system is essential in control system design due to its nonlinear, unstable, and underactuated nature, making it an ideal platform for testing various control algorithms (Fauziyah et al., 2020; Siradjuddin, Amalia, Rohadi, et al., 2018). These systems challenge nonlinearity and underactuation, serving as benchmarks for evaluating control methods (Siradjuddin, Amalia, Setiawan, et al., 2018a). The system's instability and non-minimum phase characteristics suit advanced control strategies (Banerjee & Pal, 2018; Brumand-Poor et al., 2023). Despite common use for linear control tests, the system's nonlinear behavior is evident when different pendulum responses arise from the same cart velocity, highlighting its research value in control theory. The system is a standard for control method design with applications like missile launchers and Segways, emphasizing its practical relevance (Caiado, 2012; Kuczmann, 2019; Ozana et al., 2021). It is vital to examine control strategies and advance control engineering.

The ESP32 System-on-Chip (SoC) is suitable for real-time control system implementation due to its high processing power from the dual-core Tensilica Xtensa LX6 microprocessor, facilitating complex control algorithms (Maier et al., 2017). This capability supports real-time digital proportional-integral control, as seen in advanced driving assistance algorithms for Active Front Steering and Rear Torque Vectoring (M. Choi & Choi, 2015; Her et al., 2015; Seo & Hwangbo, 2015). The SoC's low-cost, low-power attributes are ideal for embedded systems and IoT projects. Although not specifically designed for real-time applications, its features, including Wi-Fi and Bluetooth, enable wireless control systems, as shown in the portable wireless oscilloscope project (Maier et al., 2017). Wireless connectivity benefits distributed control systems or remote monitoring and control applications.

The ESP32's notable advantage is its capacity to handle multitasking, enhanced by the FreeRTOS operating system. FreeRTOS supports concurrent processes, enabling the development of responsive applications that efficiently manage multiple sensors and communication protocols. For instance, studies indicate that FreeRTOS integration on the ESP32 enhances digital output switching and analog-to-digital conversion performance, achieving high frequencies and low conversion times crucial for real-time applications (Lin & Wang, 2022; Miranda et al., 2021). This is particularly advantageous in situations that demand immediate responses, such as real-time motion detection and alerting in surveillance systems (Okokpujie et al., 2023). ESP32 microcontrollers enable real-time processing and control, while BLDC motors provide efficient and precise actuation. BLDC motors surpass traditional DC motors in cart inverted pendulum systems with higher efficiency, better speed-torque characteristics, greater dynamic response, and longer operational life (Mostafapour et al., 2015). Their compact size and high torque-to-size ratio make them ideal for space and weight-sensitive applications. Additionally, BLDC motors offer noiseless operation and broader speed ranges, beneficial which are for precise inverted pendulum control.

The Vedder Electronic Speed Controller (VESC) is a highly effective open-source motor controller for BLDC motors. Initially developed for electric skateboards, the VESC has been adapted for general robot control, providing high-speed, high-torque, and high-output capabilities (D. Choi, 2020). Its open-source nature encourages collaborative development and continuous improvement, making it a cost-effective alternative to commercial products. The VESC's efficacy is due to its advanced control algorithms and hardware upgrades. The Myongji-VESC (MJ-VESC) variant features hardware enhancements for more accurate current control and advanced position and speed control algorithms, suitable for multi-axis robot systems. This reflects the trend of sophisticated BLDC motor control techniques, such as fuzzy voltage-based controllers and adaptive PID controllers, which outperform conventional PID controllers (Sebasthirani, 2024; Vimala et al., 2019). VESC's success as an open-source motor controller for BLDC motors is due to its collaborative development, cost-effectiveness, and advanced control capabilities, making it suitable for various applications, from robotics to electric vehicles, fulfilling the need for precise and efficient motor control. The incorporation of ESP32 and BLDC motors with VESC in a cart inverted pendulum design may potentially yield more efficient and economically viable solutions for inverted pendulum systems.

This paper investigates the development of a cart inverted pendulum system utilizing ESP32 and BLDC motor technology with a VESC controller. The document is structured as follows: The subsequent section presents the mathematical model of the cart inverted pendulum, followed by a description of the hardware configuration. The ensuing section delineates the proposed software architecture, which employs freeRTOS and serial communication. An evaluation of both hardware and software components is presented in the results and discussion section. The final section concludes the study.

LITERATURE REVIEW

Mathematical Model

In control theory, the mathematical representation of the cart inverted pendulum system is a crucial field of research, particularly because of its natural instability and intricate dynamic behavior. When designing the hardware for a cart inverted pendulum, the mathematical model plays a key role in determining which sensors can be appropriately chosen to obtain the system's inputs.

The cart inverted pendulum consists of a cart that can move horizontally along a track, with a pendulum attached to it that is free to pivot about a fixed point. This system serves as an excellent benchmark for testing various control strategies due to its non-linear characteristics and the challenges associated with maintaining balance. To derive the mathematical model of the cart inverted pendulum, one commonly employs the principles of Lagrangian mechanics. The Lagrangian $\mathcal{L}$ is defined as the difference between the kinetic energy $\mathcal{T}$ and potential energy $\mathcal{V}$ of the system. The kinetic energy of the cart and pendulum can be expressed in terms of their masses and velocities, while the potential energy is related to the height of the pendulum's center of mass. The equations of motion can be derived from the Euler-Lagrange equations, leading to a set of nonlinear differential equations that describe the dynamics of the system (He et al., 2023). The state-space representation of the cart inverted pendulum is particularly useful for implementing control strategies. In this representation, the state vector typically includes the position of the cart, the angle of the pendulum, and their respective velocities. The state-space model can be expressed in the form:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Where x is the state vector, u is the control input (force applied to the cart), and A, B, C and D are matrices that define the system dynamics. This formulation allows for the application of various control techniques, such as Linear Quadratic Regulator (LQR) and PID control, to stabilize the pendulum in its upright position (Siradjuddin, Amalia, Setiawan, et al., 2018b). The control strategies for the cart inverted pendulum often focus on achieving stability through feedback mechanisms. For instance, the LQR method optimizes the control input by minimizing a cost function that penalizes deviations from desired states and excessive control efforts ((Shi et al., 2014). Research has shown that LQR can effectively stabilize the system while maintaining robustness against disturbances. Additionally, augmented PID control schemes have been demonstrated to enhance system performance by adjusting the control parameters based on the current state of the pendulum and cart. Furthermore, the cart inverted pendulum system can be characterized as a single-input, multi-output (SIMO) system, where the control input is the force applied to the cart, and the outputs are the cart's position and the pendulum's angle.

Lagrange Equation

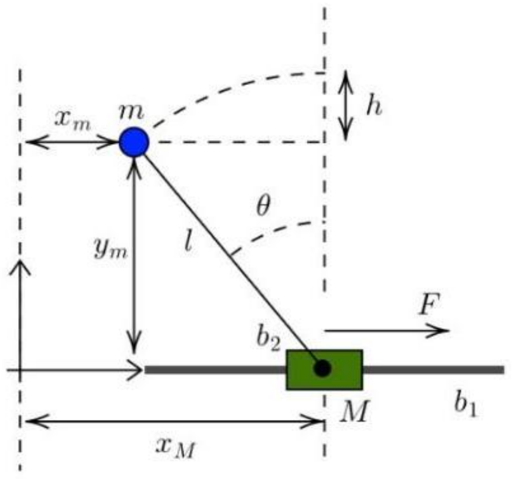


Figure 1. Cart-Inverted Pendulum Diagram
(Source : Author's Documentation, 2024)

The dynamical model of the cart-inverted pendulum is derived based on the diagram presented in Figure 1. Utilizing Lagrangian mechanics, the dynamics of the cart-inverted pendulum can be expressed as

$$(M+m)\ddot{x}_M - ml\ddot{\theta}\cos\theta + ml\theta^2\sin\theta + b_1\dot{x}_M = u$$
$$-ml\ddot{x}_M\cos\theta + ml^2\ddot{\theta} + b_2\theta - mgl\sin\theta = 0$$

Where M and m are the cart and pendulum masses, respectively. The massless pendulum rod length is denoted by l. The cart position x_M is expressed concerning the reference frame origin. θ denotes the angle of the pendulum measured around the vertical axis (the positive direction of the angle is counterclockwise). g denotes the gravity. b_1 and b_2 express the translational and rotational friction coefficients. Single and double dots are used to denote first and second derivatives, respectively. The state-space model can be derived by linearizing the dynamic model at the desired state (upright pendulum position, with zero velocities and accelerations), which can be expressed as

$$\begin{pmatrix} \dot{x}_M \\ \dot{\theta} \\ \ddot{x}_M \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mg}{M} & -\frac{b_1}{M} & -\frac{b_2}{Ml} \\ 0 & \frac{g(M+m)}{Ml} & -\frac{b_1}{Ml} & -\frac{b_2(M+m)}{Mml^2} \end{pmatrix} \begin{pmatrix} x_M \\ \theta \\ \dot{x}_M \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{M} \\ \frac{1}{Ml} \end{pmatrix} u$$

$$\begin{pmatrix} x_M \\ \theta \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_M \\ \theta \\ \dot{x}_M \\ \dot{\theta} \end{pmatrix}$$

The primary components of the cart-inverted pendulum platform design consist of a BLDC motor, an absolute encoder, an incremental encoder, an aluminum V-Slot 2040, a V-Slot gantry plate kit, belt HTD5M, and pulleys. Figure 2 depicts the front normal view of the platform. An incremental encoder AMT103 was attached to the output shaft of the gear reduction, with a reduction ratio of 15:50. The incremental encoder was utilized to measure the cart position x_M and velocity $\dot{x}_M$. Two limit switches were situated at both extremities of the V-Slot rail. The limit switches were implemented for safety purposes and cart position calibration.

Hardware Design

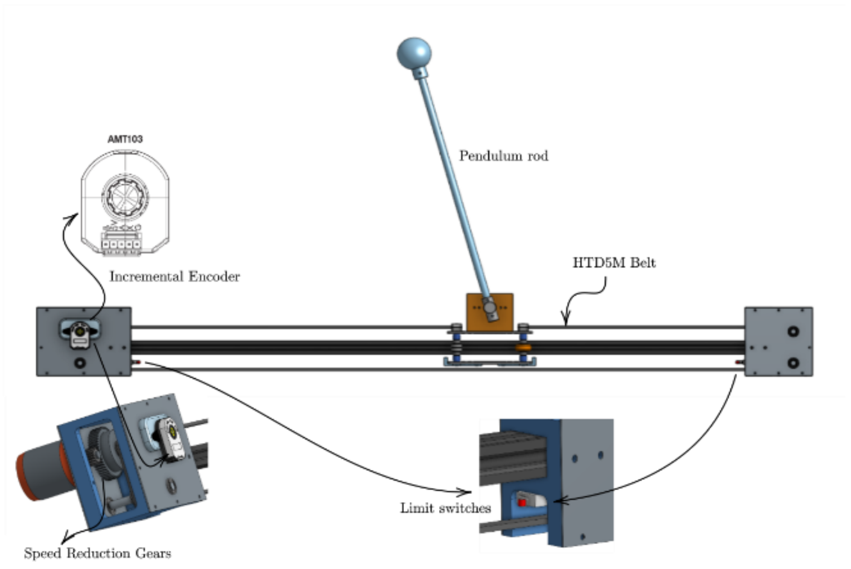


Figure. 2 Normal front view of the cart-inverted pendulum design (Source : Author's Documentation, 2024)

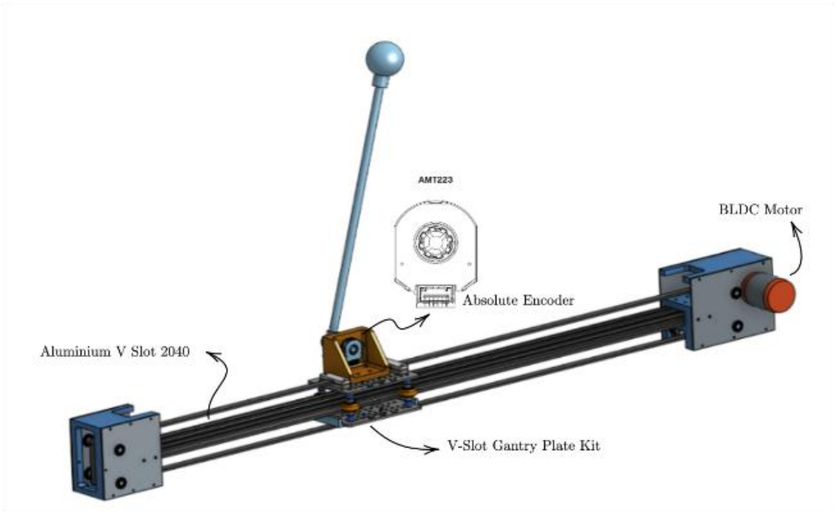


Figure 3. Isometric back view of the cart-inverted pendulum design (Source : Author's Documentation, 2024)

Figure 3 illustrates the platform isometric back view. An absolute encoder AMT223 was positioned on the center top of the V-Slot gantry plate to measure the pendulum angle θ.

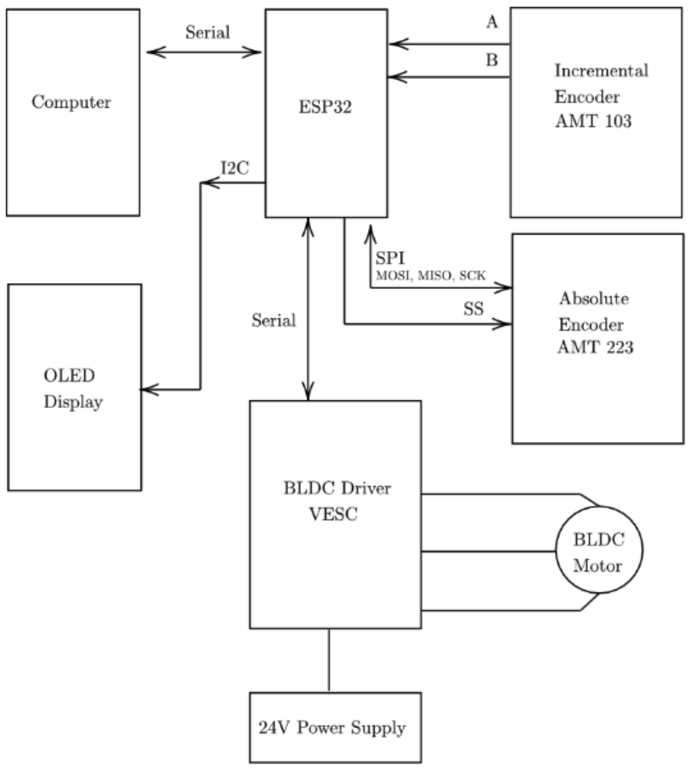


Figure 4. Circuit block diagram
(Source : Author’s Documentation, 2024)

Figure 4 illustrates the circuit block diagram of the cart inverted pendulum system. The control algorithms are executed by a computer, which communicates with an ESP32 via a serial connection to exchange sensor data and control instructions. The ESP32 interfaces with two encoders: a quadrature encoder AMT103, utilizing 2 GPIO pins connected to its A and B terminals, and an absolute encoder AMT223, employing SPI communication. To implement SPI on the ESP32, specific GPIO pins are configured for the protocol's roles, typically using four pins: MOSI, MISO, SCK, and SS. The ESP32 and VESC devices exchange commands and data through UART serial communication. A 24V DC power supply energizes the BLDC motor.

SOFTWARE DESIGN

The system's architecture consists of ESP32 firmware and a Python class library. These elements enable basic data exchanges through serial communication, allowing for the transmission of duty cycle instructions to the VESC and the collection of sensor data. Two primary serial commands are utilized:

1. "GET" is used to obtain sensor readings and limit switch states.
2. "M, X" is employed to send control commands to the VESC, where "X" represents a float value for the duty cycle. This value ranges from -1.0 to 1.0, with negative values indicating reverse motor rotation.

On the ESP32 firmware side, six freeRTOS tasks were created: readSerial, sendSerial, readAbsoluteEncoderTask, readIncrementalEncoderTask, readLimitSwitchTask, vescTask ,and writeOLED. The diagram in Figure 4 shows the connection and data flow of the firmware.

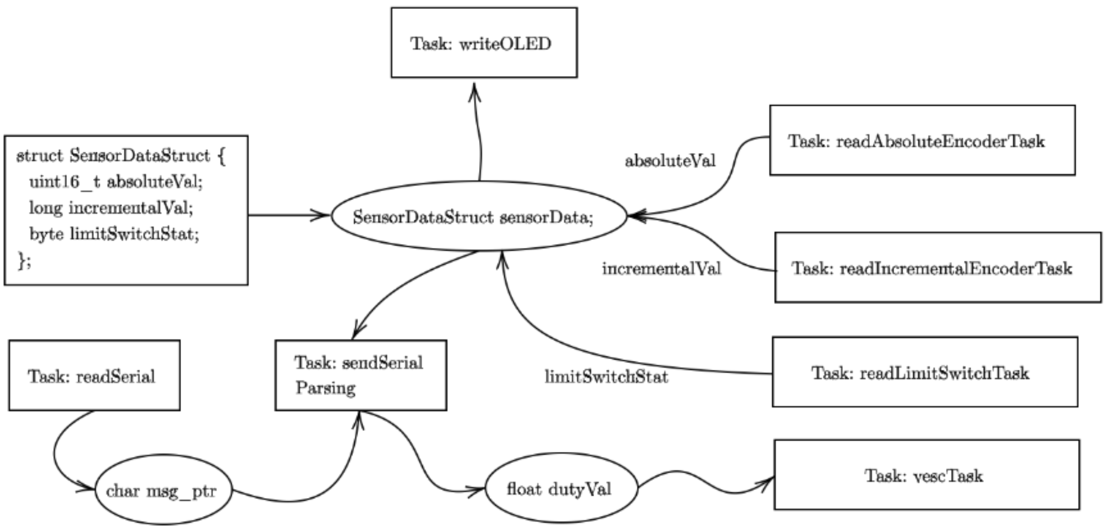


Figure 5. FreeRTOS tasks
(Source : Author’s Documentation, 2024)

The readSerial function operates by continuously monitoring and collecting serial data, while also clearing the message buffer once processing is complete. A segment of the firmware code is provided to illustrate the functionality of the readSerial task.

```
#if CONFIG_FREERTOS_UNICORE
static const BaseType_t app_cpu = 0;
#else
static const BaseType_t app_cpu = 1;
#endif
// Settings
static const uint8_t buf_len = 255;
// Globals
static char *msg_ptr = NULL;
static volatile uint8_t msg_flag = 0;
void setup() {
          Serial.begin(115200);
          xTaskCreatePinnedToCore(readSerial, "Read Serial", 4096, NULL, 1,
&readSerialHandle, app_cpu);
}
void loop() {
  // Execution should never get here
}
// Task: read message from Serial buffer
void readSerial(void *parameters) {
  char c;
  char buf[buf_len];
  uint8_t idx = 0;
  // Clear whole buffer
  memset(buf, 0, buf_len);
  // Loop forever
  while (1) {
    // Read characters from serial
    if (Serial.available() > 0) {
      c = Serial.read();
      // Store received character to buffer if not over buffer limit
      if (idx < buf_len - 1) {
        buf[idx] = c;
        idx++;
      }
      // Create a message buffer for print task
      if (c == '\n') {
        // The last character in the string is '\n', so we need to replace
        // it with '\0' to make it null-terminated
        buf[idx - 1] = '\0';
        // Try to allocate memory and copy over message. If message buffer is
        // still in use, ignore the entire message.
        if (msg_flag == 0) {
          msg_ptr = (char *)pvPortMalloc(idx * sizeof(char));
          // If malloc returns 0 (out of memory), throw an error and reset
          configASSERT(msg_ptr);
          // Copy message
          memcpy(msg_ptr, buf, idx);
          // Notify other task that message is ready
          msg_flag = 1;
        }
        // Reset receive buffer and index counter
        memset(buf, 0, buf_len);
        idx = 0;
      }
    }
  }
}
```

The input buffer's maximum length is set to 255 characters by buf_len. A pointer called msg_ptr is utilized to hold dynamically allocated message data. The volatile variable msg_flag indicates whether a new message is ready for processing, with 1 signifying readiness and 0 indicating otherwise. The readSerial task is created and assigned to a specific CPU core using xTaskCreatePinnedToCore, with parameters including the task function, name, stack size (4096 bytes), parameters (NULL), priority (1), and handle (&readSerialHandle). When msg_flag is 0, indicating no current message processing, memory for msg_ptr is dynamically allocated using pvPortMalloc. The configASSERT() function checks for successful memory allocation, triggering an error and system reset if unsuccessful. The message is then transferred from buf to msg_ptr, and msg_flag is set to 1 to indicate a new message's readiness. Finally, the buffer (buf) is cleared using memset, and idx is reset to 0 in preparation for the next message.
The computer transmitted a command through a serial connection, which was then verified using a parsing algorithm. This algorithm was incorporated into the sendSerial task function. The code for the sendSerial function follows.

```
void sendSerial(void *parameters) {
  while (1) {
    // Wait for flag to be set and print message
    if (msg_flag == 1) {
      // Serial.println(msg_ptr);
      // Parse command and value
      char *token = strtok(msg_ptr, ",");
      if (token != NULL) {
        strcpy(command, token);
        String cmd = String(command);
        cmd.trim();
        cmd.toUpperCase();
        token = strtok(NULL, ",");
        if (token != NULL) {
          value = atof(token);  // Convert to float
          if (cmd == "Z") {
            setZeroSPI(cs_pin);
            vTaskDelay(50 / portTICK_PERIOD_MS);
          } else if (cmd == "M") {
            dutyVal = float(value);
          }
        }
        if (cmd == "GET") {
          Serial.printf("sensor,%u, %ld, %u\n",
                        sensorData.absoluteVal,
                        sensorData.incrementalVal,
                        sensorData.limitSwitchStat);
        }
        vTaskDelay(2 / portTICK_PERIOD_MS);  // Adjust delay as necessary
      }
      vPortFree(msg_ptr);
      msg_ptr = NULL;
      msg_flag = 0;
    }
  }
}
```

The `sendSerial` function operates as a FreeRTOS task, designed to interpret serial commands received by an Arduino ESP32. These commands are stored in the global `msg_ptr` variable, which is populated by the `readSerial` function. The task initiates by verifying if `msg_flag` is set to 1, signaling that a new message is available for processing. The incoming message is then segmented using `strtok(msg_ptr, ",")`, separating it based on commas. The initial segment is expected to be the command, with subsequent segments potentially containing related parameters or values. Upon successful extraction of a segment, it is copied into a `command` array using `strcpy`. To ensure uniform command handling, the command string undergoes trimming and is converted to uppercase, making it case-insensitive. The next segment is extracted as the expected value using `strtok(NULL, ",")`. If a value is present, it's converted to a float using `atof(token)`. The task then responds to specific parsed commands (`cmd`): For "Z", it invokes `setZeroSPI(cs_pin)`, which zeros the value of an absolute encoder controlled via SPI. A 50-millisecond delay is implemented using `vTaskDelay` to allow the operation to complete. For "M", it assigns the received float `value` to a global variable `dutyVal`, adjusting the output duty cycle for a BLDC controller through VESC. For "GET", it transmits a formatted string back to the serial console, containing sensor data including fields like `absoluteVal`, `incrementalVal`, and `limitSwitchStat`. After processing, the memory allocated for the message (`msg_ptr`) is freed using `vPortFree()` to prevent memory leaks. Finally, `msg_ptr` is set to `NULL`, and `msg_flag` is reset to 0, indicating that the message has been processed and the system is ready for the next command.

The following codes are readLimitSwitchTask and vescTask function implementations.

```
// Task to read limit switch 1
void readLimitSwitchTask(void *pvParameters) {
  while (true) {
    leftSwitch = digitalRead(leftSwitch_pin);
    rightSwitch = digitalRead(rightSwitch_pin);
    sensorData.limitSwitchStat = 0b00000000;
    bitWrite(sensorData.limitSwitchStat, 0, leftSwitch);
    bitWrite(sensorData.limitSwitchStat, 1, rightSwitch);
    vTaskDelay(1 / portTICK_PERIOD_MS);
  }
}
void vescTask(void *pvParameters) {
  while (true) {
    VescUART.setDuty(dutyVal);
    //vTaskDelay(1 / portTICK_PERIOD_MS);
  }
}
```

The VescUART.h library enables the management of BLDC motors controlled by a VESC through UART communication, allowing for motor speed adjustments via the setDuty() function. VESC employs a serial communication protocol, typically over UART, to accept instructions and setup parameters from a microcontroller like the ESP32. It utilizes a specific packet format that incorporates command types and related data to regulate various motor attributes, including speed, current, and duty cycle. The ESP32 leverages its built-in UART capabilities for data transmission and reception, operating at a specified baud rate (often 115200 for VESC). The UART pins (such as TX and RX) are linked to the VESC, enabling two-way communication. The writeOLED task displays continuously the sensor states which can be obtained from the data struct named SensorDataStruct. The following piece of code describes the implementation of the writeOLED function and SensorDataStruct declaration.

```
// Define the sensor data structure
struct SensorDataStruct {
  uint16_t absoluteVal;  // Value from the absolute encoder
  long incrementalVal;   // Value from the incremental encoder
  byte limitSwitchStat;  // Status of the limit switches
};

// Global variable to hold sensor data
volatile SensorDataStruct sensorData;

// Write the sensor data OLED
void writeOLED(void *pvParameters) {
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  display.display();
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  // Clear the buffer
  display.clearDisplay();
  display.setTextSize(1);               // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);  // Draw white text
  display.setCursor(0, 0);
  display.println("Hello World");
  display.display();
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  for (;;) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("Sensor Data");
    display.print(sensorData.absoluteVal);
    display.print(", ");
    display.print(sensorData.incrementalVal);
    display.print(", ");
    display.println(sensorData.limitSwitchStat);
    display.println("Incoming Command");
    display.print(command);
    display.print(", ");
    display.println(value);
    display.display();
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}
```

RESULTS AND DISCUSSION

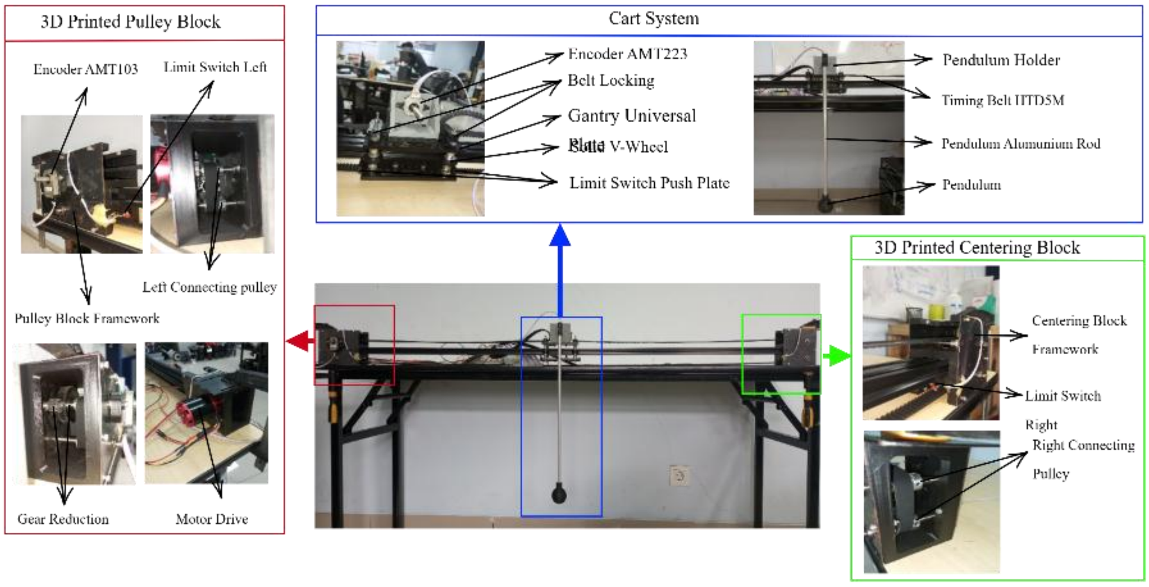


Figure 6. Mechanical System of Cart Inverted Pendulum
(Source : Author's Documentation, 2024)

Figure 6 shows the realization of the design of the tool that has been created to support this research. There are several mechanical parts of the designed inverted pendulum cart, which consist of 3D Printed Pulley Block, cart, and 3D Printed Centering Block. The 3D Printed Pulley Block frame is made of a 1.5 mm iron plate, and PLA+. On the 3D Printed Pulley Block Frame, there is also a gear reduction to improve performance, namely the M1 15T and 50T spur gears. The input gear reduction is connected to the N5065 5065 270kv Brushless Motor and the output gear reduction will be one shaft with the timing of the HTD5M 10T pulley which will be installed in a circle on the 3D Printed Centering Block and cart as many as 4 pieces.

The cart line uses an Aluminum slot profile 2040 V-slot. The Cart uses 2 pieces of 40 mm Universal Plate Gantry which is connected by bolts and support nuts along with V-wheel solid wheels that will be connected to aluminum slot profiles. The pendulum mount is made of PLA+ which is given a Flanged F688zz bearing and an 8mm stainless rod as the pendulum arm shaft so that the pendulum can move freely until it reaches the inverted position. The connection of the stainless rod of the pendulum shaft and the aluminum sleeve of the pendulum rod is connected with the PLA+ part. The cart also has an HTD5M belt lock that connects 3 parts made of PLA+.

The 3D Printed Centering Block section has a main frame made of PLA+ and iron plates as well as free bearing and timing pulleys that function as a counterweight to the 3D Printed Pulley Block. These components are designed to ensure stability, strength, and accuracy during operation. This designed component is also equipped with the need for nut holes and locking bolts that will connect the sensor and parts. In addition, the mechanical system is assisted by locking on the 3D Printed Pulley Block and 3D Printed Centering Block to connect the table and cart inverted pendulum so that the cart inverted pendulum does not experience a shift in the cart inverted pendulum.

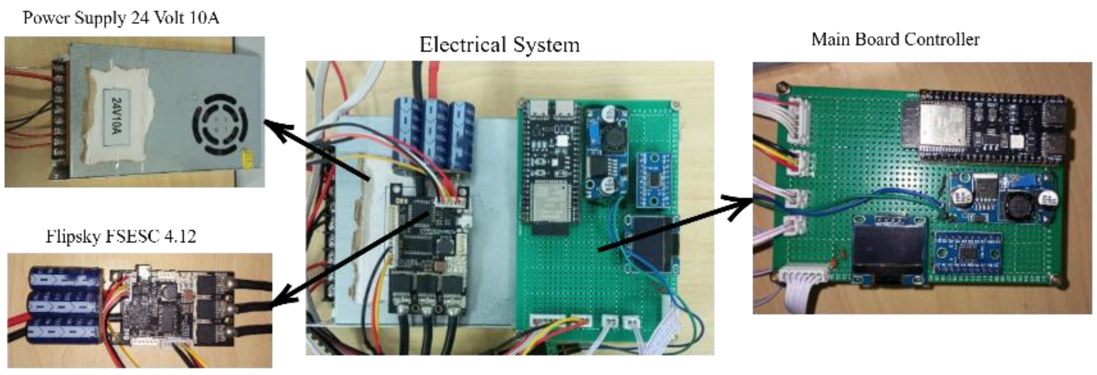


Figure 7. Electrical System of Cart Inverted Pendulum
(Source : Author's Documentation, 2024)

Figure 7 shows the electrical system that will support the operation of the appliance. This electrical system is made to control the inverted pendulum cart. The components used are designed to work together efficiently to maintain the balance of the pendulum and control the movement of the cart. The output voltage is connected to the input from the BLDC driver motor and the step-down voltage to meet the voltage required in the microcontroller, sensor, and voltage converter.

In this system, an actuator in the form of a BLDC N5065 270kv motor is used as a driving force. To control the motorcycle, a motorcycle driver called an Electronic Speed Controller (ESC) is used, namely FSESC Flipsky 4.12. The Communication pin is connected to the ESP32-S3 with the UART Serial using the RX, TX, and GND pins. The 3-phase FSESC output is connected to a 270kv N5065 BLDC motor. The ESC will process the read sensor data and will communicate the data through the communication pins. The ESC will also receive the microcontroller feedback to command the BLDC motor.

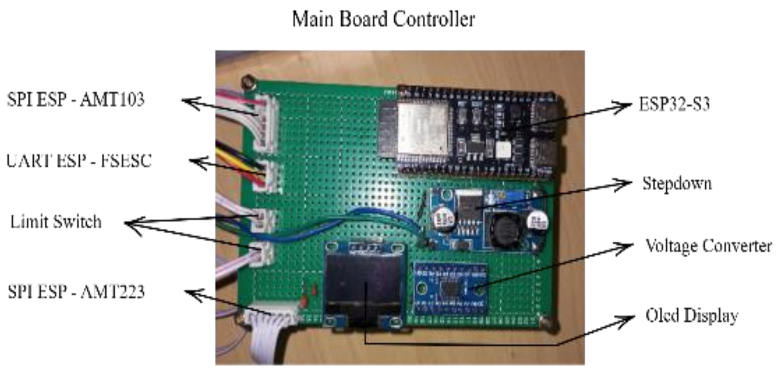
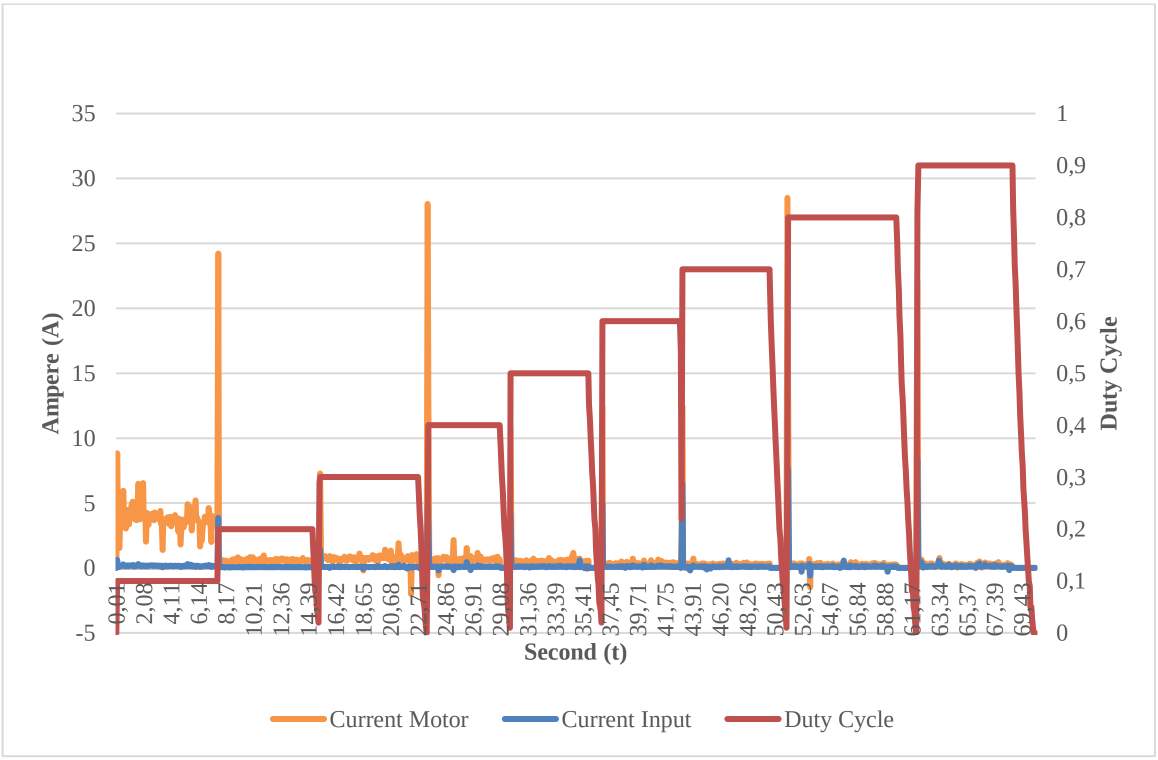


Figure 8. Main Board System of Cart Inverted Pendulum
(Source : Author's Documentation, 2024)

The main board controller is a main board controller of the system, where all commands are executed and processed. In this mainboard system, there is the brain of the system, namely the ESP32-S3 microcontroller. The ESP32-S3 will process the data read by the AMT223, AMT103, and limit switch sensors. The ESP 32 pins used are GPIO 10, GPIO 11, GPIO 12, GPIO 13, VCC, and GND pins connected to the AMT103 sensor pins to measure the rotational speed of the motor. GPIO 9, GPIO8, VCC and GND pins are connected to OLED displays. The GPIO 5 and VCC pins are connected to the left limit switch and the GPIO 4 and VCC pins are connected to the right limit switch used as a buffer or emergency switch.

The AMT223 encoder sensor is used to detect angles on the pendulum shaft. The encoder sensor is connected to a voltage converter which is used as a voltage converter from 5V (encoder output) to 3.3V (ESP32-S3 input). This system voltage converter is used because the ESP32-S3 operates at a logic voltage of 3.3V, so a level shifter is required for signal compatibility. The wiring circuit is designed using PCB boards to integrate components, ensuring the flow of data and energy runs efficiently. In addition, the appliance is equipped with a 24 Volt 10A DC Power Supply power source that provides stable energy. The power requirements of the components used in the circuit PCB vary between 3.3 – 5 Volts, therefore the LM2596 stepdown is used. Communication between the sensor and the ESP32-S3 is regulated through the SPI communication protocol, while FSESC and ESP communication uses Serial UART which allows the tool to work automatically and in real-time.

To improve the precision of the cart in balancing the pendulum, the ESP32-S3 microcontroller must generate a PWM signal with a duty cycle that matches the control algorithm. To integrate FSESC data with ESP32 there are several commands namely "vesc.getVescValues()". PWM signals with the appropriate duty cycle will be used by the motor driver to control the power provided to the motor. The duty cycle is the percentage of signal time at a "high" condition for a full period that has been set. The duty cycle will affect current and ERPM. The highest duty cycle value set is 90%. The experiment of the effect of the duty cycle percentage can be seen in figure 9 below.



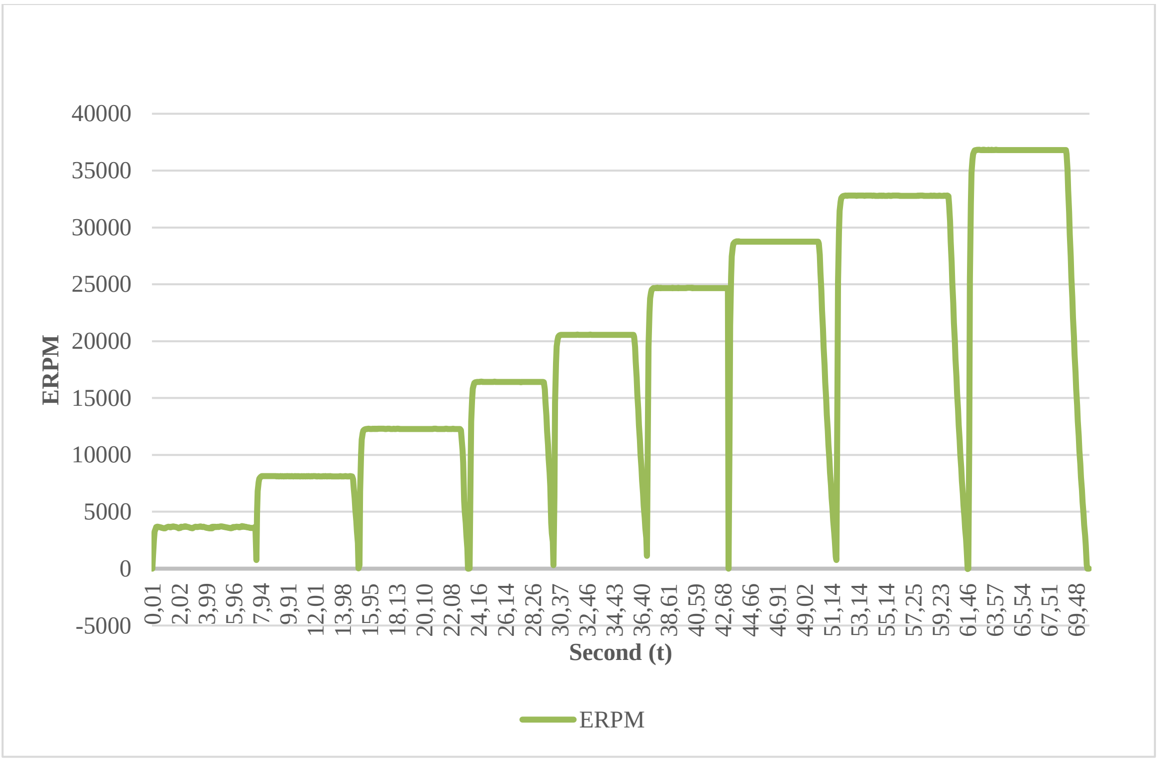


Figure 9. Graph Effect of Duty Cycle Percentage on Current and ERPM. (Source : Author's Documentation, 2024)

From the experiment on the influence of the duty cycle on current and ERPM values in BLDC motors above, it can be seen that the duty cycle value has an effect on the initial inlet current input to the motor. The input current of the motor will experience a surge at the beginning of the start and then it will be stable. The current value is lower than that of the current motor, which shows the suitability of the power conversion theory in VESC. The current value of the motor is greater than the current in, especially when the duty cycle is high. This is because the duty cycle increases the average voltage of the motor, so that the motor current rises to meet the required torque. This graph shows that each increase in the duty cycle affects the amount of current in and motor current. When the duty cycle is close to maximum, both the current in and the current motor reach its peak.

The effect of duty cycle value on ERPM (Electrical Revolutions Per Minute) can be seen in Figure 9. From the graph above, it can be seen that the ERPM value will increase linearly with the duty cycle, which shows that the average stress to the motor (generated by the duty cycle) directly affects the rotational speed of the motor. This is by the characteristics of BLDC motors where the RPM is proportional to the average voltage applied.

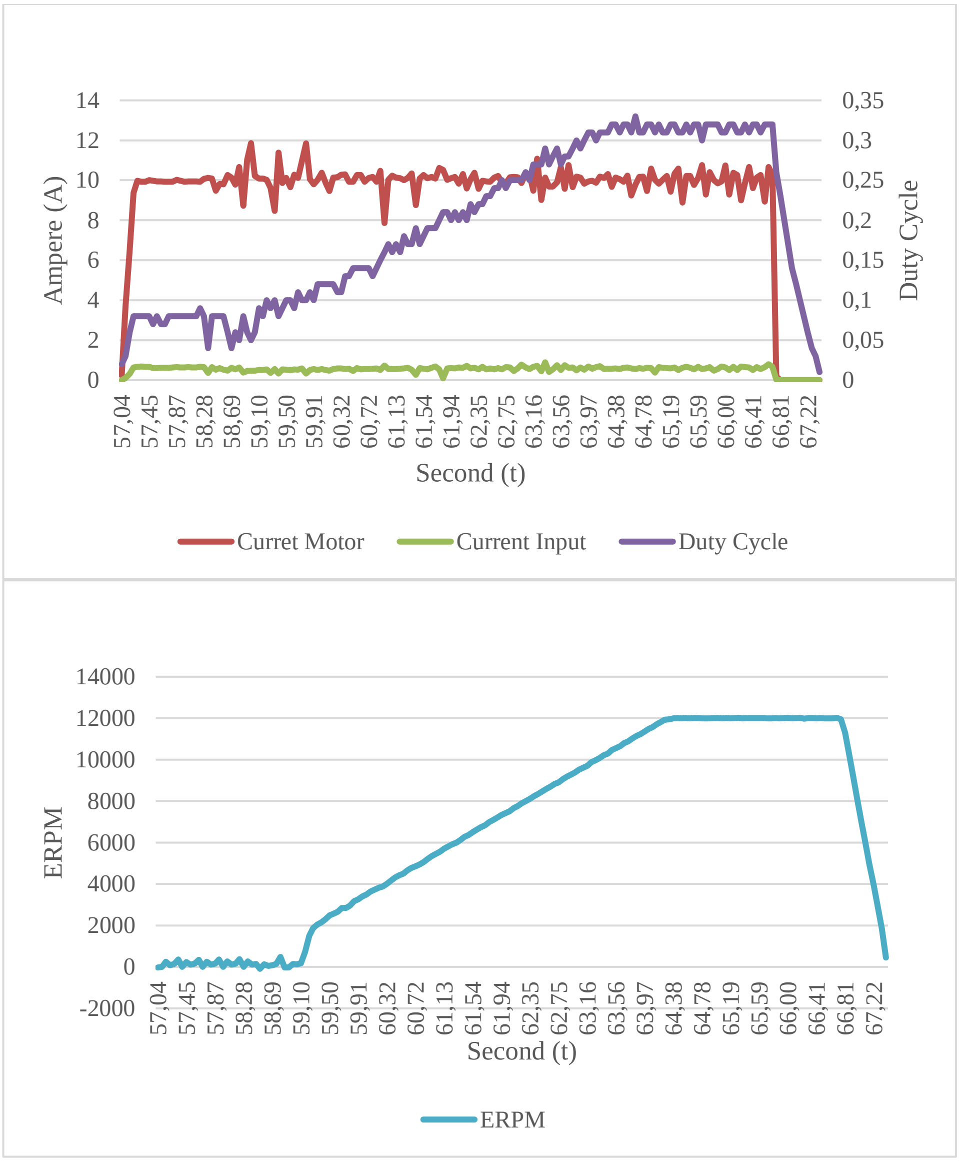


Figure 10. Graph of the Effect of Continuous Duty Cycle on Currentand ERPM. (Source : Author's Documentation, 2024)

To find out the effect of the duty cycle further on continuous conditions, it can be seen in Figure 10. It can be seen that a continuous duty cycle will affect the value of the inrush current on the motor and will experience a surge. Then there is a stable position without a drastic decrease in current as well as ERPM and voltage. So from this condition it can look better, especially for the service life of BLDC motorcycles because there is very little voltage and current drop.

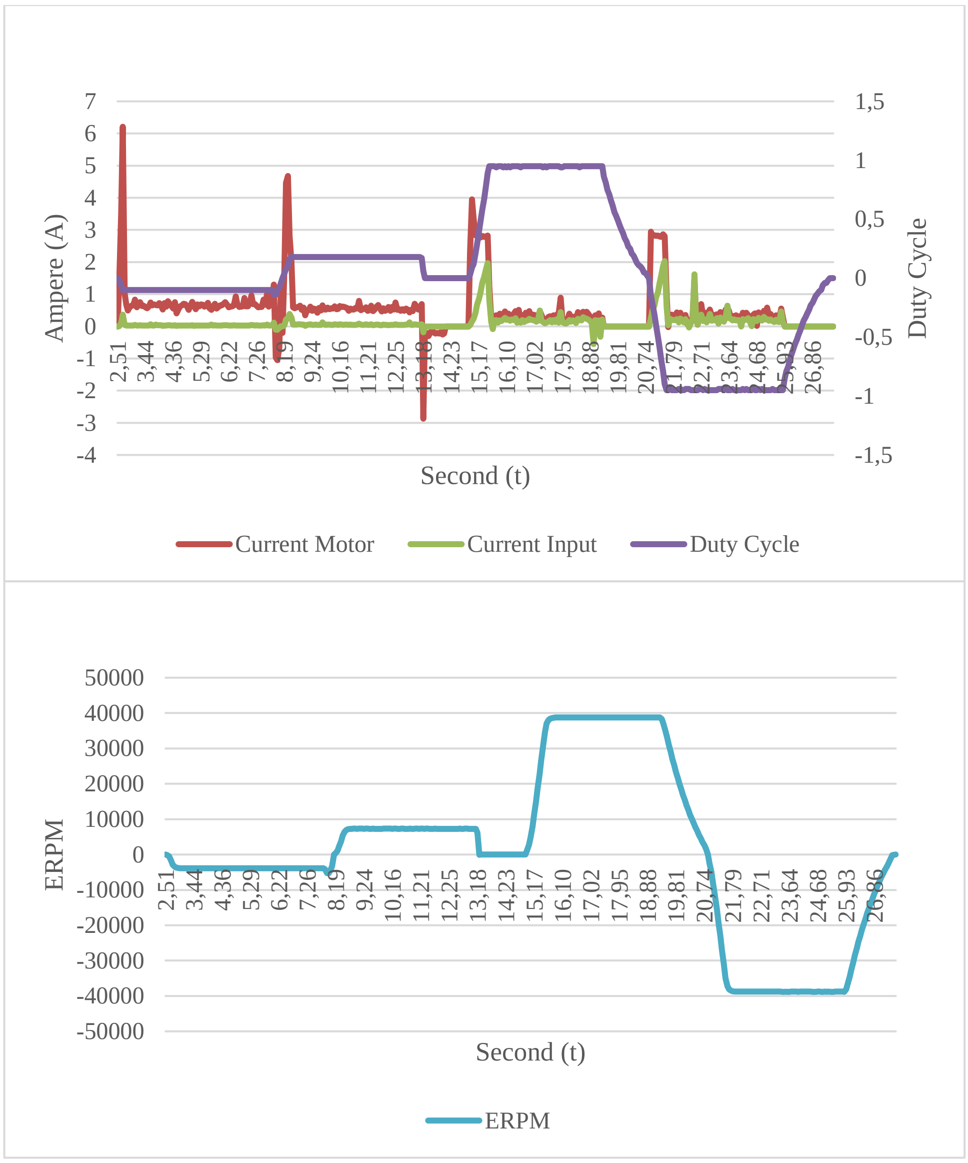


Figure 11. Graph Cycle Duty CW and CCW on Currentand ERPM. (Source : Author's Documentation, 2024)

Figure 11 is a duty cycle experiment on a BLDC motor with CW (clockwise) and CCW (counter clock wise) movements. On the basis of theory, controlling the direction of rotation of the motor can be done by controlling the commutation system. Where on the ESC driver there are transistors that will commutate alternately. If the motor is desired in the opposite direction or CCW then the transistor will be controlled to commute in reverse order. To reverse the commutation, the Duty cycle sent to the ESC is a positive value for CW and negative for CCW. So in Figure 11, it can be seen that a positive duty cycle produces a positive ERPM, while a negative duty cycle produces a negative ERPM (the motor rotates in the opposite direction). When the duty cycle returns to zero, the ERPM slowly decreases, which may be the result of mechanical friction or the influence of the load.

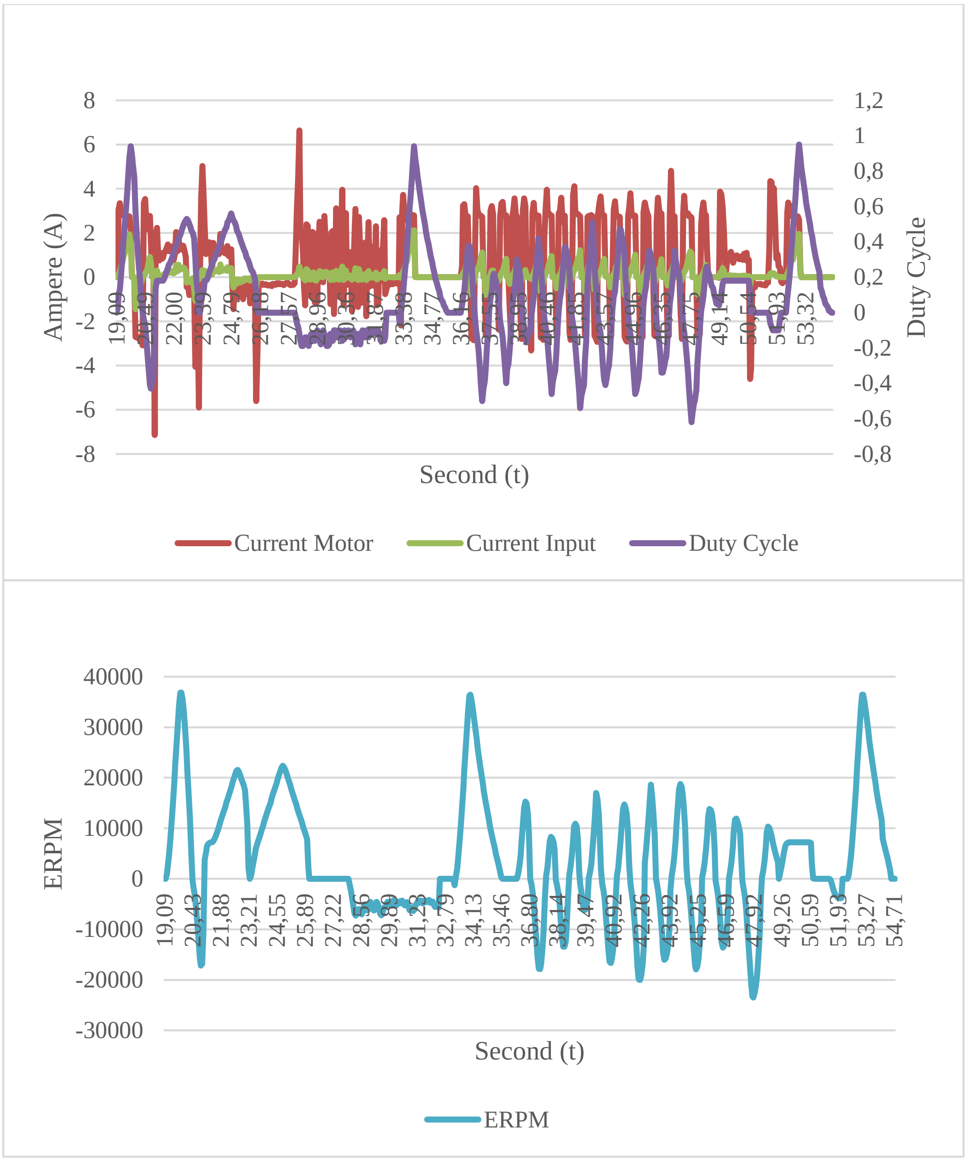


Figure 12. Graphs Duty Cycle CW and CCW with Fast Time
(Source : Author's Documentation, 2024)

Figure 11 is a duty cycle experiment on a BLDC motor with CW (clockwise) and CCW (counter clock wise) movements. Based on theory, controlling the direction of rotation of the motor can be done by controlling the commutation system. Where on the ESC driver there are transistors that will commutate alternately. If the motor is desired in the opposite direction or CCW then the transistor will be controlled to commute in reverse order. To reverse the commutation, the Duty cycle sent to the ESC is a positive value for CW and negative for CCW. So in Figure 11, it can be seen that a positive duty cycle produces a positive ERPM, while a negative duty cycle produces a negative ERPM (the motor rotates in the opposite direction). When the duty cycle returns to zero, the ERPM slowly decreases, which may be the result of mechanical friction or the influence of the load.

CONCLUSION

It has been shown that BLDC motors with VESC provide efficient and precise actuation for cart inverted pendulum systems. VESC can control accurate current, position, and speed through signal communication and input. The validity and advantages of the integration of ESP32 and BLDC motors with VESC in the design of the cart inverted pendulum have been verified with electrical, mechanical, and software systems.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to the industry collaboration research grant provided by the State Polytechnic, which supported this research project. This funding has been invaluable in facilitating our work and enabling the successful completion of our research.

REFERENCES

Journal Articles

Brumand-Poor, F., Kauderer, L., Matthiesen, G., & Schmitz, K. (2023). Application of deep reinforcement learning control of an inverted hydraulic pendulum. *International Journal of Fluid Power*, 393-418.

Caiado, M. I. (2012). Controllability of a planar inverted pendulum on a cart. *arXiv preprint arXiv:1203.4533*.

Fauziyah, M., Amalia, Z., Siradjuddin, I., Dewatama, D., Wicaksono, R. P., & Yudaningtyas, E. (2020). Linear quadratic regulator and pole placement for stabilizing a cart inverted pendulum system. *Bulletin of electrical engineering and informatics*, *9*(3), 914-923.

He, J., Li, Y., Wei, Z., & Huang, Z. (2023). Gain-Scheduled Model Predictive Control for Cart–Inverted-Pendulum with Friction and Disturbances. *Applied Sciences*, *13*(24), 13080.

Kuczmann, M. (2019). Comprehensive survey of PID controller design for the inverted pendulum. *Acta Technica Jaurinensis*, *12*(1), 55-81.

Mostafapour, J., Reshadat, J., & Farsadi, M. (2015). Improved rotor speed brushless DC motor using fuzzy controller. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, *3*(2), 78-88.

Okokpujie, K., Okokpujie, I. P., Young, F. T., & Subair, R. E. (2023). Development of an Affordable Real-Time IoT-Based Surveillance System Using ESP32 and TWILIO API. *Journal homepage: http://iieta. org/journals/ijsse*, *13*(6), 1069-1075.

Sebasthirani, K., Maruthupandi, P., Manimegalai, M., Silambarasan, N., Vannan, R. M., & Venkatesh, C. (2024). Implementation of AI Tuned PID Controller for Speed and Direction Control of BLDC Motor. *Journal of Electrical Systems*, *20*(7s), 2221-2228.

Seo, Y. W., & Hwangbo, M. (2015). A computer vision system for lateral localization. *Journal of Field Robotics*, *32*(7), 1004-1014.

Shi, X., Xu, Z., Tian, K., & He, Q. Y. (2014). Optimal Control for Wheeled Inverted Pendulum Based on Collaborative Simulation. *Applied Mechanics and Materials*, *556*, 2444-2447.

Siradjuddin, I., Amalia, Z., Rohadi, E., Setiawan, B., Setiawan, A., Putri, R. I., & Yudaningtyas, E. (2018). State-feedback control with a full-state estimator for a cart-inverted pendulum system. *International Journal of Engineering & Technology*, *7*(4.44), 203-209.

Vimala, P., Balamurugan, C. R., Subramanian, A., & Vishwanath, T. (2019). Optimization and comparative analysis of PID and FOPID controller for BLDC motor. *IAES International Journal of Robotics and Automation*, *8*(3), 174.

Proceedings

Banerjee, R., & Pal, A. (2018, December). Stabilization of inverted pendulum on cart based on lqg optimal control. In *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)* (pp. 1-4). IEEE.

Choi, D. (2020). Development of open-source motor controller framework for robotic applications. *IEEE Access*, *8*, 14134-14145.

Choi, M., & Choi, S. B. (2016). MPC for vehicle lateral stability via differential braking and active front steering considering practical aspects. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *230*(4), 459-469.

Her, H., Joa, E., Yi, K., & Kim, K. (2016). Integrated chassis control for optimized tyre force coordination to enhance the limit handling performance. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *230*(8), 1011-1026.

Maier, A., Sharp, A., & Vagapov, Y. (2017, September). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In *2017 Internet Technologies and Applications (ITA)* (pp. 143-148). IEEE.

Miranda, B. D., de Oliveira, R. S., & Carminati, A. (2021, July). Analysis of FreeRTOS Overheads on Periodic Tasks. In *Anais do XX Workshop em Desempenho de Sistemas Computacionais e de Comunicação* (pp. 119-130). SBC.

Ozana, S., Docekal, T., Nemcik, J., Krupa, F., & Mozaryn, J. (2021, June). A Comparative Survey Of Software Computational Tools In The Field Of Optimal Control. In *2021 23rd International Conference on Process Control (PC)* (pp. 284-289). IEEE.

Siradjuddin, I., Amalia, Z., Setiawan, B., Ronilaya, F., Rohadi, E., Setiawan, A., Rahmad, C & Adhisuwignjo, S. (2018). Stabilising a cart inverted pendulum with an augmented PID control scheme. In *MATEC Web of Conferences* (Vol. 197, p. 11013). EDP Sciences.

Website

Lin, C. K., & Wang, B. Y. (2022). Analyzing FreeRTOS Scheduling Behaviors with the Spin Model Checker. *arXiv preprint arXiv:2205.07480*.