SMART FAN CONTROL SYSTEM

Isha Sharma

Final Project Report
ECEN 5613 Embedded System Design
May 06, 2023

1 INTRODUCTION

The smart fan control system prototype is a sophisticated electronic device that has been designed to control the speed and direction of a DC motor-powered fan using an STM32 microcontroller and a potentiometer. The system's key objective is to provide users with a convenient and efficient means of regulating the airflow and temperature of a room through the manipulation of the fan's speed.

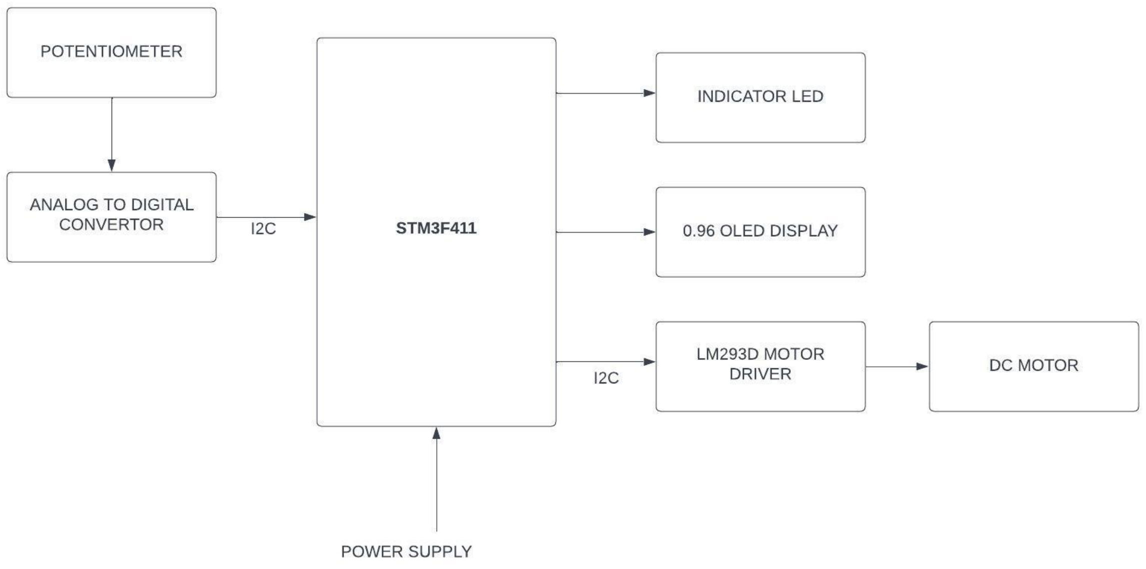
The motivation behind this project was to gain a deep understanding of the intricacies involved in creating an electronic device that functions like the everyday fans we all use at home, with the added benefit of new and exciting features like a display. The system's design and construction aim to showcase the possibilities of integrating modern technology into the traditional appliances we use daily and the potential benefits of such integrations.

The system comprises several vital components, including a potentiometer, a DC motor, an STM32 microcontroller, and a motor driver. The potentiometer serves as an input device for adjusting the speed of the fan, while the STM32 microcontroller reads the potentiometer's voltage output and uses this information to regulate the fan's speed.

The DC motor is connected to the STM32 microcontroller through a motor driver, which enables the microcontroller to control the motor's speed and direction. By modifying the voltage and polarity of the DC motor, the microcontroller can easily manage the fan's speed and direction.

The system's added feature of a display serves to provide users with essential feedback on the fan's current speed and other important parameters.

a. 1.1 BLOCK DIAGRAM



The above block diagram shows the flow of the hardware setup. The potentiometer is connected to an external 16-bit ADC. The output of the ADC is used for processing by the microcontroller based on which the DC motor speed is controlled by the driver. The proceed output is also used to turn the LED off/on. The OLED display is used to have a user interface for the smart on/off and its speed.

2 TECHNICAL DESCRIPTION

This chapter includes the hardware, firmware, software, and testing process used to develop the prototype.

2.1 Hardware Description

The prototype hardware consists of the Potentiometer+ADC circuit, indicator LED circuit, L293D+DC motor circuit, and OLED circuit. The new hardware that were interfaced:

B. ADS1115 ANALOG-TO-DIGITAL CONVERTOR

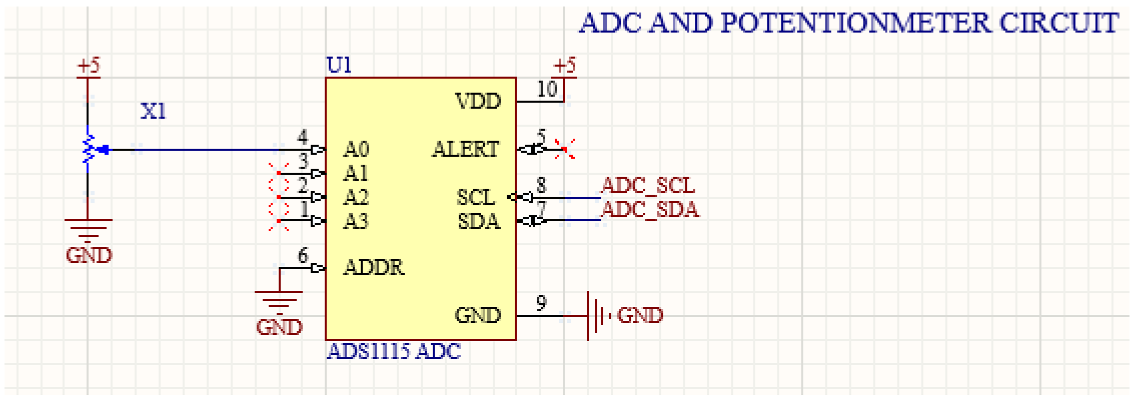
Analog Devices, Inc. has developed the AD1115, which is a 16-bit analog-to-digital converter (ADC) that is designed to deliver outstanding accuracy and resolution for precise analog signal measurement in various applications. It employs a successive approximation register (SAR) architecture, enabling it to achieve 16-bit resolution while consuming minimal power, making it highly suitable for applications that demand high precision and low power consumption. SAR divides the analog signal into a series of discrete steps, which are then compared to a reference voltage to determine the digital output value. It uses a binary search method to determine the correct digital output value with high precision and accuracy.

AD1115 features an internal voltage reference, which provides a stable and precise voltage reference for the conversion process. The input signal range can be configured through software to accept either unipolar or bipolar inputs, and it can handle a maximum input voltage range of ±VREF, typically at 2.048V. Moreover, an external voltage reference can be connected to applications that require a different reference voltage. It comes with a programmable gain amplifier (PGA) that amplifies the input signal by up to 128 times, making it highly suitable for applications where the input signal is weak or low amplitude. The PGA can be configured through software to adjust the gain and input voltage range, which adds to the ADC's flexibility in various applications.

It can communicate with the microcontroller through an I2C interface, which allows for easy integration with a wide range of microcontrollers and digital systems. The ADC supports a maximum sampling rate of 860 samples per second, and its low power consumption makes it ideal for use in battery-powered applications.

Its low power consumption, programmable gain amplifier, advanced diagnostic features, and programmable power-down mode enhance the ADC's functionality and ensure high reliability and accuracy with the potentiometer interface. The I2C/SPI interface option provides added convenience and compatibility.

The AD1115 was selected as a component of choice for the smart fan control system because of its low power consumption, high accuracy, and versatility. The ADC's programmable gain amplifier, advanced diagnostic features, and programmable power-down mode enhance its functionality and ensure high reliability and accuracy with the potentiometer interface. The option of using either the I2C or SPI interface adds to the ADC's easy-to-use feature.



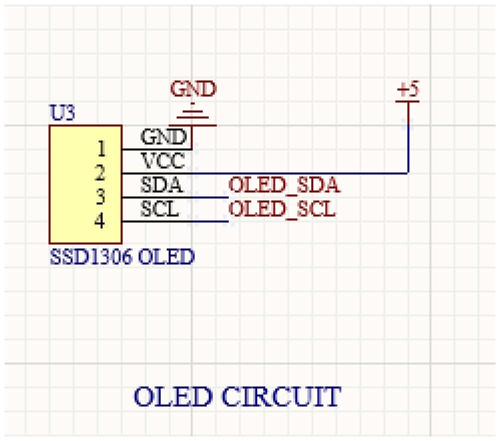
The schematic above shows the hardware connections between the potentiometer, ADC, and the microcontroller. The ADC SCL and SDA pins are the ones used for communication between the ADC and the microcontroller board. The Analog read channel 0 was used to configure the potentiometer. ADDR was grounded to have a slave address of 1001000.

C. SSD1306 128 x 64 OLED

The SSD1306 is a 128 x 64 organic light-emitting diode (OLED) display driver. The display driver is designed to provide a clear and high-contrast display, which is well-suited for use in various applications, including wearable devices, portable gadgets, and smart home appliances. It features a direct display RAM (DDRAM) that allows for fast and efficient data transfer, which results in a smooth and responsive display. The DDRAM is organized in a 128 x 64-bit array and allows for both horizontal and vertical addressing modes, which enables flexible data access and display control. The SSD1306 also includes a character generator RAM (CGRAM) that allows users to define custom characters and icons, which can be displayed on the OLED screen. The CGRAM is organized in a 64 x 8-bit array and can store up to eight user-defined characters or icons. The OLED display driver communicates with the microcontroller through either a Serial Peripheral Interface (SPI) or an I2C interface, which allows for easy integration with a wide range of microcontrollers and digital systems. The SSD1306 features a built-in charge pump that boosts the input voltage to provide a high-brightness output.

The SSD1306 OLED display is an I2C slave device, which means that it is controlled by the microcontroller acting as the I2C master. The microcontroller sends commands and data to the SSD1306 over the I2C bus, specifying the position of each pixel, the brightness of each pixel, and other display parameters. To communicate over I2C, the SSD1306 has two pins, SDA (Serial Data) and SCL (Serial Clock). The microcontroller sends data to the SSD1306 by toggling the SCL line to clock the data on the SDA line. The data is sent in packets or frames, with each frame including an address byte and one or more data bytes. The microcontroller sends commands and data to the SSD1306 using a specific set of commands that are defined in the SSD1306 datasheet. These commands include instructions to set the display on or off, set the contrast level, set the display start line, and set the display memory addressing mode. Once the microcontroller has sent the necessary commands and data to the SSD1306, the OLED display controller will update the display accordingly. The updated display information is stored in the internal RAM of the SSD1306, which can be refreshed as necessary to provide a smooth and responsive display.

The reasons for choosing this component were majorly the cost, availability, high-contrast display, fast and efficient data transfer, and custom character and icon support.



The hardware connections majorly consist of the SCA and SCL pins that are used to communicate with the microcontroller.

D. L239D Motor Driver IC

The L293D is a popular and versatile motor driver IC that is commonly used for interfacing with DC motors. The IC is designed to provide bidirectional control for two DC motors or a single stepper motor, making it an excellent choice for a variety of motor control applications.

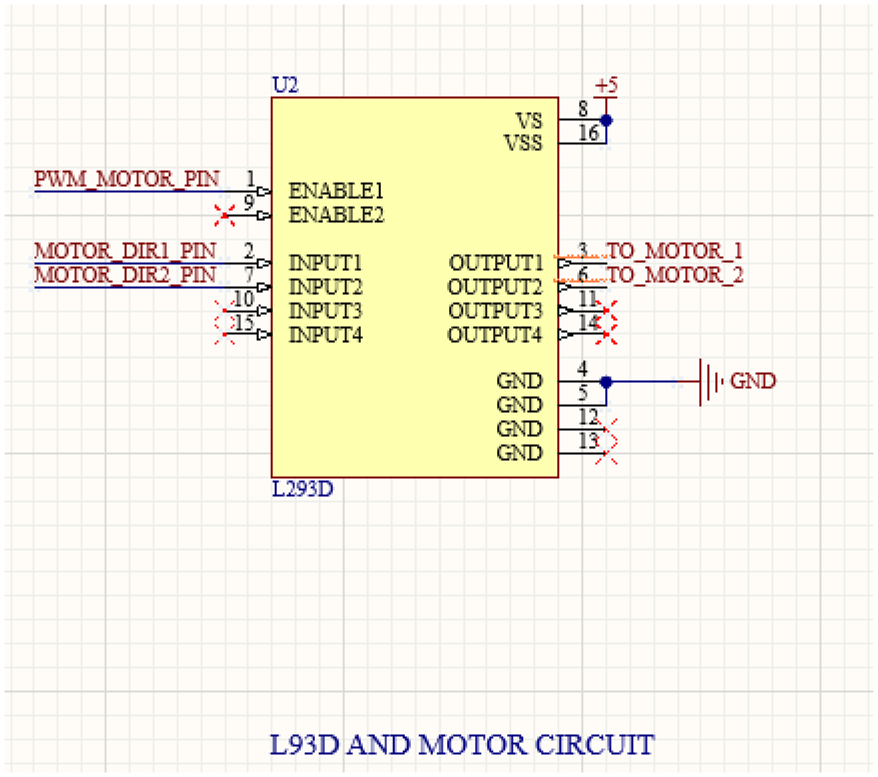
The L293D consists of two H-bridge circuits that enable it to control the direction and speed of a DC motor. Each H-bridge is composed of four transistors and diodes, which provide bidirectional current flow and prevent damage to the IC due to voltage spikes. Each H-bridge is composed of four transistors and diodes that provide bidirectional current flow and prevent damage to the IC due to voltage spikes. The H-bridge circuit consists of four switches, which can be either transistors or MOSFETs, that are connected in a specific configuration to control the current flow to the motor. When the switches are configured correctly, the current can be directed through the motor in either direction, allowing the motor to spin in either direction.

It also includes a control logic circuit that receives signals from a microcontroller or other external control circuitry. The control logic circuit interprets these signals to control the switches in the H-bridge circuits and thereby controls the direction and speed of the motor.

The L293D motor driver IC can handle a maximum operating voltage of 36V and a maximum continuous current of 600mA per channel, making it suitable for driving small to medium-sized DC motors. It also features thermal overload protection, which prevents damage to the IC due to overcurrent or overheating.

To interface the L293D with a DC motor, the motor is connected to the output pins of the H-bridge, while the control pins are connected to the microcontroller or external control circuitry. The control pins are used to set the direction and speed of the motor by sending signals to the IC.

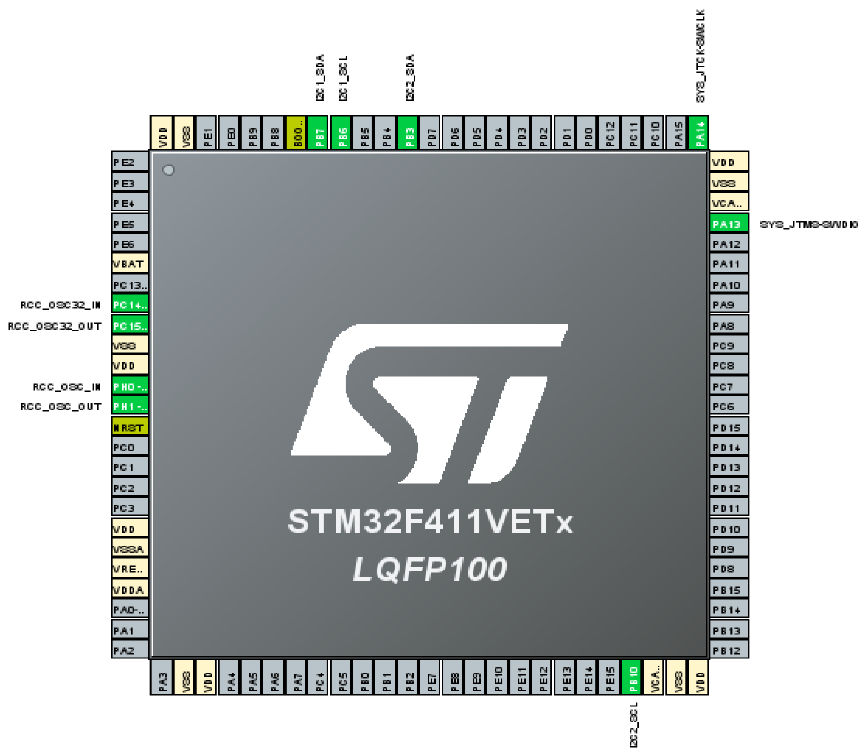
Reasons for choosing this component were its ability to provide bidirectional control, handle moderate currents and voltages, protect voltage spikes, and protect against thermal overload making it a reliable and efficient choice for driving DC motors.



The PWM output of the microcontroller is fed to enable the L293D IC. 2 GPIOS of the microcontroller are configured to control the direction of the motor.

2.2 Firmware Design

Firmware for the smart fan control system consists of files for each part. The screenshot below shows the .ioc of my project. It labels the pins based on I2C1/2 and other pins required for the firmware design.



The firmware can be split into 4 parts: Indicator LED interface, OLED interface, Potentiometer+ADC interface, and the DC motor+L293D interface.

A. Indicator LED interface

Steps to interface an LED at A5 pin:

Register	Function
RCC_AHB1ENR	enabling clock to GPIOA
GPIOA_MODER	setting the pin as output
GPIOA_BSRR	set or reset the pin to turn LED on/off

The BSRR was used to turn the led on/off based on the potentiometer readings.

B. OLED interface

The OLED was interfaced using the I2C1 of the Stm32 board. The following were configured in the .ioc file to set up the I2C communication

RCC-> High-speed clock: Crystal/Ceramic Resonator
I2C1 -> Speed: Fast mode
I2C1 -> Clock Speed:400000 Hz
SYS-> Debug: Serial Wire

The below table shows the different functions to interface the OLED.

Function	**Description**
SSD1306_WRITECOMMAND	Writes command to the I2C address
ssd1306_I2C_Write	writes a single byte to the slave
ssd1306_I2C_WriteMulti	writes multiple bytes to the slave
SSD1306_OFF	turns OLED off
SSD1306_ON	turns OLED on
SSD1306_Clear	Clears screen
SSD1306_Init	initializes the OLED using the commands in the datasheet, resets variables
SSD1306_UpdateScreen	updates buffer from internal RAM to OLED
SSD1306_Fill	Fills the entire OLED with either OLED color or to blank the screen
SSD1306_DrawPixel	draws a pixel at a desired location
SSD1306_GotoXY	sets the cursor pointer at desired x and y location
SSD1306_Putc	puts a character in internal RAM
SSD1306_Puts	puts a string in internal RAM
SSD1306_ScrollRight	scrolls the display to right
SSD1306_ScrollLeft	scrolls the display to left
SSD1306_Stopscroll	stops scrolling the display

The files define the necessary macros and variables used by the functions mentioned above.
I am thankful to Alexander Lutsai and Tilen Majerle for their online resources that helped me build my code.

A separate file was not made for the interfacing of ADC and instead, it was done directly in the main.

C. Potentiometer + ADC interface

The ADS1115 was interfaced using the I2C2 of the Stm32 board. The following were configured in the .ioc file to set up the I2C communication.

RCC-> High-speed clock: Crystal/Ceramic Resonator
I2C1 -> Speed: Fast mode
I2C1 -> Clock Speed:400000 Hz

SYS-> Debug: Serial Wire

The bus can operate at fast mode for 400kHz clock frequency.

Steps to interface ADC:
1) set slave address based on addr connection. In my case, it is 1001000 because addr is grounded.
2) Write to config register based on the following given below:

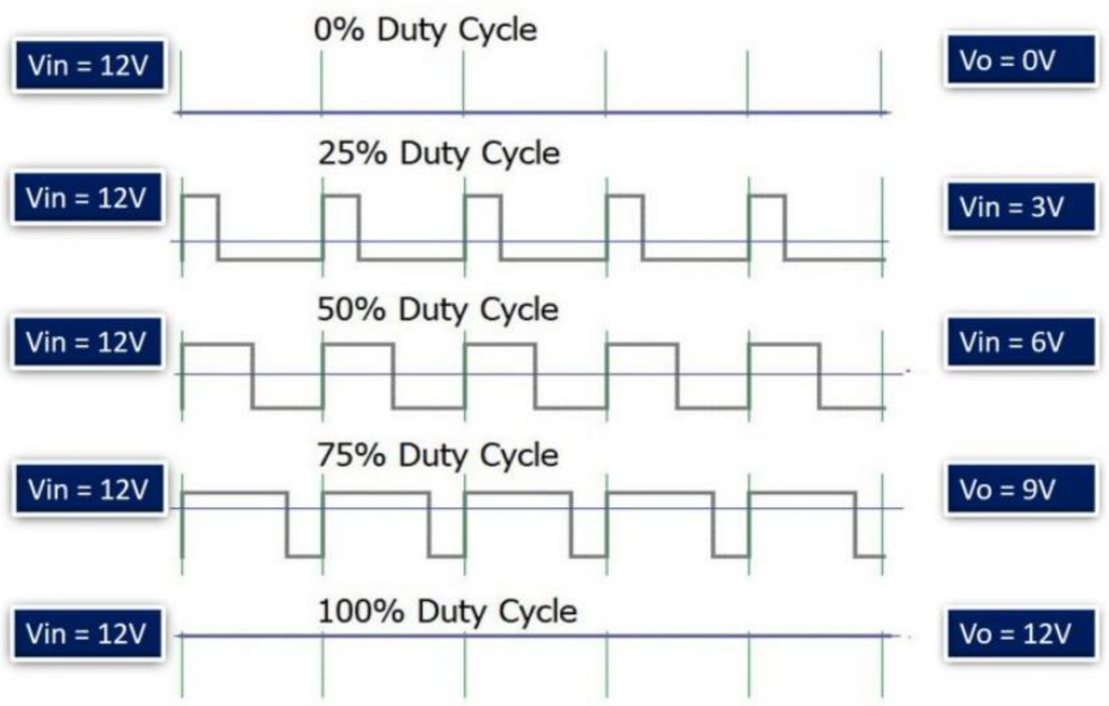
Register fields	**Function**
OS	1: single conversion
MUX	100: compare with AN0 and GND
PGA	000: FSR =+- 6.144 V
MODE	1: single shot mode
DR	100: default
COMP	00011: all default

3) Read data from the conversion register and save it into a variable.

4) Conversion of received analog reading into voltage is done based on the PGA register. For my case, since the voltage was above 4v, the conversion factor is 6.114/32768.

D. L293D + DC motor interface

A dc motor's speed can be controlled using pulse width modulation. The direction of the motor can be controlled by changing the polarity of the input source. The L293D ic uses the H-bridge for the same function. An H-Bridge consists of four switches, resistors, and two control signals as shown in the figure. Two logic probes control the direction of a DC motor. When the left probe logic signal is on, the motor rotates in an anti-clockwise direction and when the right probe logic signal is on, the motor rotates in a clockwise direction.



The L293D enable pin for the dc motor requires a PWM output. The steps to generate the PWM using TIM4 are:

1. Initialise PB9 as the pwm output pin

Register	**Function**
RCC_AHB1ENR	enabling clock to GPIOB
GPIOB_MODER	setting the pin as alternate function output
GPIOB_AFR	setting alternate function to TIM channel 4
GPIOB_OTYPER	set output as push-pull
GPIOB_OSPEEDR	setting output speed as high
GPIOB_BSRR	setting initial output as low

2. Following the same steps above, initialize 2 GPIO pins (PB12, PB13) for motor direction control as output and set PB12 to high and PB13 to low to rotate in the clockwise direction.
3. Initialize the TIM4 for pwm output:

Register	**Function**
RCC_AHB1ENR	enabling clock to TIM4

TIM4_PSC	setting the prescalar value for 48MHz as 4799
TIM4_ARR	setting auto-reload register value as 999 for 10 kHz frequency
TIM4_CCR4	setting duty cycle
TIM4_CCMR2	setting pwm mode 1 as output
TIM4_CCER	enabling output for channel 4
TIM4_CR1	enable counter for TIM4

4. the duty cycle is updated every time with the converted voltage value from the ADC
5. Hence, based on the potentiometer reading, the duty cycle for the dc motor changes which finally changes the speed of the motor rotation.

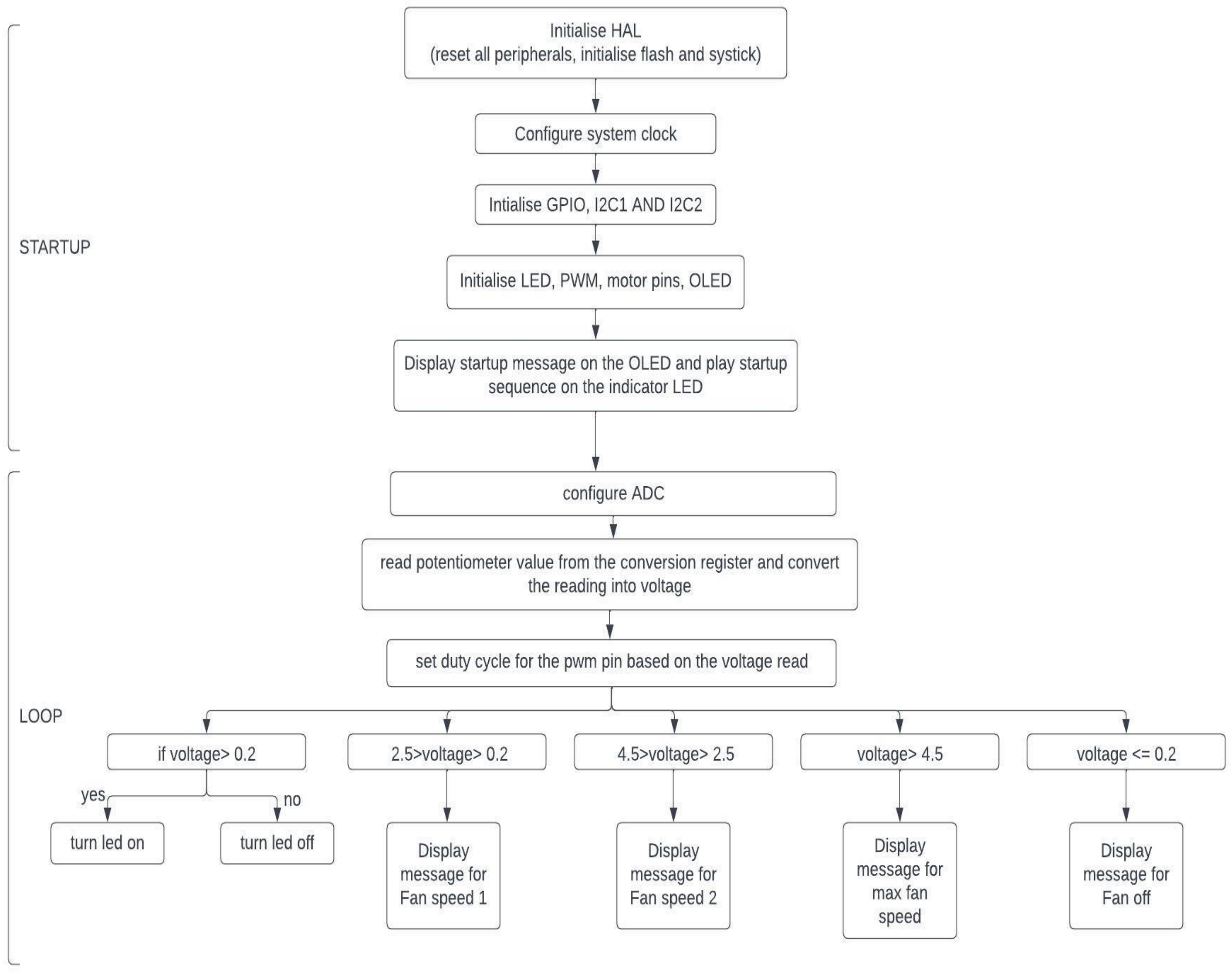
2.3 Software Design

The software used for the development of the code is STMCubeIDE. It is an Integrated Development Environment (IDE) designed by STMicroelectronics for programming and debugging STM32 microcontrollers. It provides a complete software development platform that includes code generation, compilation, debugging, and flashing tools in a single environment. STMcubeIDE is based on the Eclipse framework and features a user-friendly graphical interface that allows developers to create and manage their projects easily. It also includes a wide range of software libraries, drivers, and examples that enable rapid prototyping and development of complex applications.

A big chunk of the code was done using the Hardware Abstraction Layer library. The HAL library is a key feature of the STM32CubeIDE software. It is a high-level library that provides a consistent API for accessing various hardware peripherals on the STM32 microcontroller, regardless of the specific model or series. This allows developers to write portable and reusable code, without having to worry about the low-level details of the hardware.
The HAL library abstracts away the register-level programming required to access the hardware peripherals, making it easier and faster to develop embedded applications. It provides a set of generic APIs that can be used to control various hardware peripherals such as GPIOs, UARTs, I2C, SPI, timers, ADCs, and many more. The HAL library also provides an easy-to-use code-generation tool called "CubeMX". CubeMX allows developers to quickly configure their hardware peripherals and generate initialization code for their projects. This saves time and eliminates the need for manual configuration of registers, which can be error-prone.
The reason to use the HAL library was that it simplified the process of developing applications by providing a consistent API for accessing hardware peripherals and a code generation tool for configuring it.

The software design for the smart fan control system involves multiple modules that work together to achieve the desired functionality. The main module is responsible for initializing and configuring the peripherals used in the system, such as the ADC, GPIO, and timers. Another module is responsible for reading the analog input from the potentiometer using the ADC and scaling it to the required range for the fan speed control. The system also includes a module to control the DC motor using the L293D motor driver, which receives its control signals from the STM32 microcontroller's GPIO pins. The system also includes an OLED module that displays information such as the current fan speed and system status. Additionally, the system includes an indicator LED module to indicate the on/off status of the fan.

The software design algorithm for the smart fan control system is as follows:


STARTUP
Initialise HAL (reset all peripherals, initialise flash and systick)
Configure system clock
Intialise GPIO, I2C1 AND I2C2
Initialise LED, PWM, motor pins, OLED
Display startup message on the OLED and play startup sequence on the indicator LED
LOOP
configure ADC
read potentiometer value from the conversion register and convert the reading into voltage
set duty cycle for the pwm pin based on the voltage read
if voltage> 0.2
yes
no
turn led on
turn led off
2.5>voltage> 0.2
Display message for Fan speed 1
4.5>voltage> 2.5
Display message for Fan speed 2
voltage> 4.5
Display message for max fan speed
voltage <= 0.2
Display message for Fan off

2.4 Testing Process

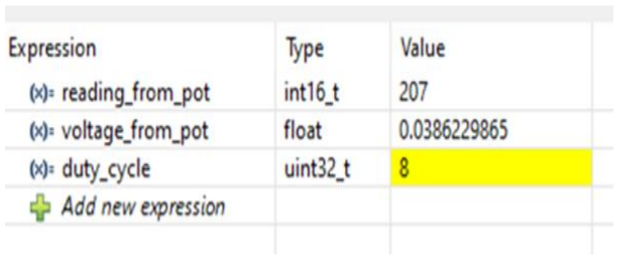
Since I was building the project on my own, the testing process for the project was very informal and manual. To ensure the system functioned correctly, I implemented a systematic testing approach. I conducted regression testing to ensure that any changes or updates to the system did not impact previously working features. Each hardware component's testing was done using a multimeter. The code for each component was written first and tested. All of them were combined and the whole system was tested to work as intended.

3 RESULTS AND ERROR ANALYSIS

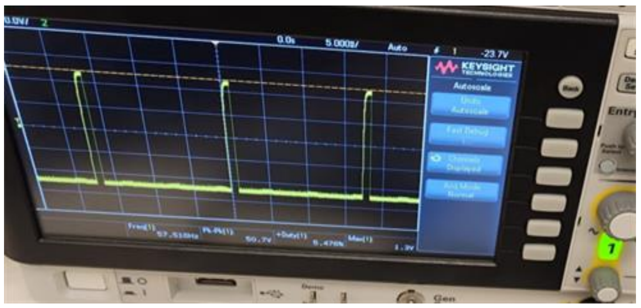
For reference to the images and observations shown below, the following variables were used:
1) reading_from_pot: variable that shows the raw value read by the ADC of reading from the potentiometer (range: 0-25457)
2) voltage_from_pot: variable that shows the converted voltage value of the reading (range: 0.0-4.75)

3) duty_cycle: variable that shows the current duty cycle calculated based on the voltage for PWM output (range:0-1000)

Case 1: FAN OFF

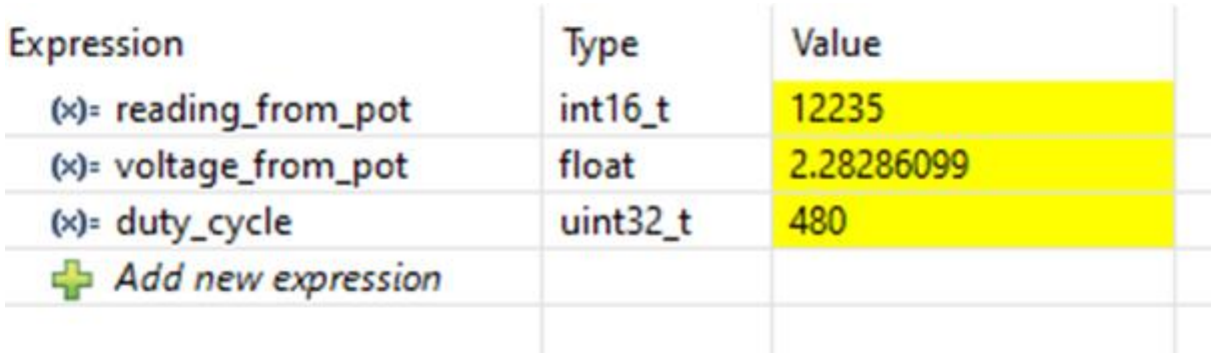


Expression	Type	Value
(x)= reading_from_pot	int16_t	207
(x)= voltage_from_pot	float	0.0386229865
(x)= duty_cycle	uint32_t	8
Add new expression		



The above two images show the first case of the system which is when the potentiometer is turned to the lowest possible position. The reading from the pot and the voltage converted are not exactly 0 and hence the duty cycle, as also shown in the oscilloscope, is approximately a min of 4-5%.
This must be because the potentiometer was old and maybe a little too tight for it to rotate fully.

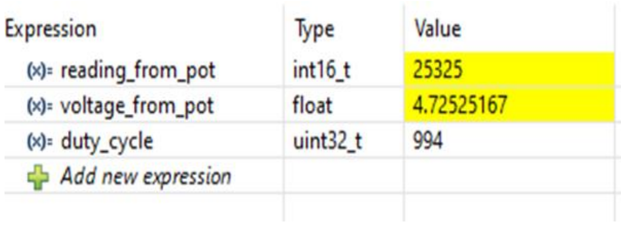
Case 2: FAN MID SPEED



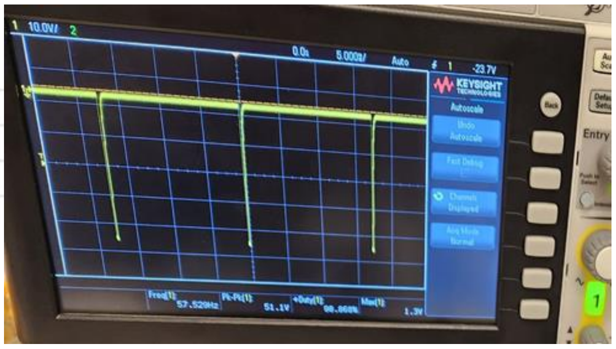
Expression	Type	Value
(x)= reading_from_pot	int16_t	12235
(x)= voltage_from_pot	float	2.28286099
(x)= duty_cycle	uint32_t	480
Add new expression		

The above image shows the second case of the system which is when the potentiometer is turned to the mid position. The reading from the pot and the voltage converted are not approximately mean values and hence the duty cycle is approximately a min of 480 which is approximately 50%.

Case 3: FAN MAX SPEED



Expression	Type	Value
(x)= reading_from_pot	int16_t	25325
(x)= voltage_from_pot	float	4.72525167
(x)= duty_cycle	uint32_t	994
Add new expression		



The above two images show the last case of the system which is when the potentiometer is turned to the maximum possible position. The reading from the pot and the voltage converted are approximately maximum and hence the duty cycle, as also shown in the oscilloscope, is approximately a max of 99.4%.

Problems faced:
Throughout the course of the project, a critical technical issue was encountered with the L293D output pins, as they were found to have a peak-to-peak voltage output of 40+ volts. This presented a significant challenge since the DC motor being utilized in the project was designed to function on a mere 5V. Consequently, the motor failed to operate as expected, leading to a substantial setback in the project's progress. Several troubleshooting techniques were employed, including a thorough check of wiring connections and a detailed review of the datasheets. However, the issue persisted, and it was ultimately determined that the PWM output at the L293D pins was not compatible with the DC motor. As a result, the motor was dropped.

Key Learnings:
Undertaking this project independently provided me with a plethora of technical skills and knowledge. One of the most significant aspects of the project was gaining expertise in interfacing GPIOs, including configuring and manipulating the input/output of various pins. Another essential skill developed through this project was the ability to work with the I2C protocol, which facilitated communication between different components of the system. Additionally, I learned how to interface with an OLED display, configure Pulse Width Modulation (PWM), and work with Analog Digital Conversion (ADC) - all crucial technical skills. Coding in Bare Metal and using HAL were some skills that needed a lot of my attention and have given me confidence.

The project also helped me to hone my problem-solving abilities and to analyze problem statements critically to arrive at appropriate solutions. Debugging errors within a limited time constraint also tested my critical thinking and problem-solving skills. The project also required me to think about a problem from a product perspective, considering factors such as usability and user experience.

One of the most important takeaways from the project was improving my time management skills, which continue to benefit me in my future projects and professional endeavors. In summary, this project provided an opportunity for me to develop a range of technical, interpersonal, and problem-solving skills that will prove invaluable to my future endeavors.

4 CONCLUSION

I have always been fascinated by how embedded systems are hidden in our day-to-day lives. One of the reasons that I chose this project was to get my hands dirty with such systems. Developing a smart fan was something that helped me understand how embedded systems are used to solve problems in our daily lives.

Another reason I chose this project was motivated by my desire to reinforce my foundational understanding of embedded systems. By working on this project, I had the opportunity to gain knowledge about various concepts such as ADCs, communication protocols, smart displays, fundamental driver circuits, bare metal coding, HAL coding, and other related topics. The practical experience I gained through this project enabled me to strengthen my skill set in the basics of embedded systems, providing me with a solid foundation for future projects in this field.

A smart fan system is a useful and efficient way to regulate fan speed and temperature in a variety of settings. The use of an OLED display and intuitive controls makes it easy to use. It serves as an example of the potential of embedded systems in enhancing everyday life and provides valuable lessons for future projects.

5 FUTURE DEVELOPMENT IDEAS

There are several future development ideas for the smart fan control system that can enhance its functionality and efficiency. One such idea is to incorporate voice control, allowing users to adjust the fan speed and temperature using voice commands. This would add a new level of convenience and accessibility to the system, making it ideal for use in situations where the user's hands may be occupied.

Another development idea is to integrate the smart fan system with other smart home devices, such as thermostats or air purifiers, to create a comprehensive home automation system. This would enable the system to work seamlessly with other devices to provide a comfortable and healthy living environment.

In addition, the smart fan system could be further optimized to reduce energy consumption and increase energy efficiency. For instance, it could be programmed to adjust the fan speed automatically based on ambient temperature or humidity levels, reducing the need for manual adjustments and ensuring optimal energy usage.

Moreover, incorporating advanced sensors such as air quality sensors or motion sensors could enable the smart fan system to adapt to the user's environment and provide more personalized and responsive control. This would make the system more intelligent and efficient in regulating temperature and air quality.

Additionally, a temperature sensor can be used to monitor the fan speed, replacing the need for a potentiometer. A system that can approximately count the number of people in the room and control the airflow accordingly can also be used.

The smart fan control system can also be used in industries/ factories to regulate the temperature and airflow based on the requirements.

Furthermore, the smart fan control system can also be integrated with IoT (Internet of Things) technologies, enabling remote control and monitoring via smartphones or other devices. This would add more flexibility and convenience, allowing users to adjust fan settings and receive alerts from anywhere, anytime.

Finally, the smart fan system could be designed to have a more aesthetically pleasing appearance, with sleek and modern designs that can blend seamlessly into any environment. This would make the system more attractive to users and could encourage wider adoption of the technology in homes and workplaces.

Overall, the possibilities for future development and optimization of the smart fan control system are endless, and there is great potential for the technology to continue to evolve and improve in the years to come.

6 ACKNOWLEDGEMENTS

This project would not have been possible without the support and help of many people.

I would like to express my sincere gratitude to Prof. Linden McClure for his constant support and direction throughout the course of this project.

I would also like to thank the TAs: Maanas, Saanish, and Jordi for their unwavering support, guidance, and suggestions that were instrumental in shaping my ideas and improving my work.

A special thanks to my classmates and friends for their continuous support and encouragement throughout the project. There were moments during the semester and while working on this project when I felt demotivated, but their persistent reminders and motivation kept me on track and helped me complete this project successfully.

A small token of my appreciation goes to my parents who motivated me to keep working hard and not focus on the insignificant things/problems around me.

I would also like to thank the authors of various websites and resources that I referred to while working on this project. Their work was immensely helpful in gaining a deeper understanding of the concepts and techniques involved in this project. Without their contributions, completing this project would have been much more challenging.

7 REFERENCES USED

[1] https://www.youtube.com/watch?v=M5ddTjrcvEs&t=54s
[2] https://microcontrollerslab.com/dc-motor-l293d-motor-driver-ic-arduino-tutorial/
[3] https://microcontrollerslab.com/l293d-motor-driver-ic-introduction-pinouts-example/
[4] https://www.youtube.com/watch?v=4jcxeJxvi3Y
[5] https://www.ti.com/lit/ds/symlink/ads1115.pdf
[6] https://www.digikey.com/htmldatasheets/production/2047793/0/0/1/SSD1306.pdf
[7] https://thecodeprogram.com/ads1115-with-stm32-cubemx
[8] https://cdn-shop.adafruit.com/datasheets/l293d.pdf
[9] STM32F411VE Datasheet, Reference manual, and User Manuals
[10] https://www.thomasnet.com/articles/instruments-controls/types-of-motor-controllers-and-drives/
[11] https://microcontrollerslab.com/analog-to-digital-adc-converter-working/
[12] https://deepbluembedded.com/stm32-pwm-example-timer-pwm-mode-tutorial/
[13] https://deepbluembedded.com/?s=dc+motor
[14] https://controllerstech.com/oled-display-using-i2c-stm32/
[15] Professor, TAs and Classmates

E. APPENDICES

Several appendices have been attached to this report in the order shown below.

7.1 Appendix - Bill of Materials

Part Description	**Source**	**Cost**
ADS1115 16 Bit ADC	Amazon www.amazon.com	$7.99
SSD1306 I2C 128X64 OLED	Amazon www.amazon.com	$7.29
L293D Stepper Motor Driver	Amazon www.amazon.com	$8.99
Resistor	Embedded Systems Lab	$1.00
LED	Already had it	$0.00
Jumpers	Already had it	$0.00
STM32F411E Board	Embedded Systems Lab	$0.00
TOTAL		**$25.27**

Note: If I were doing this project over again, I would have obtained from L293D Stepper Motor Driver alternate source as it arrived pretty late.

7.2 Appendix - Schematics

7.3 Appendix - Source Code