BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 (July 2023)

Microprocessor and Embedded Systems Laboratory

Final Project Report

Section: A1 Group: 01

STM32 Blue Pill Dev Board

Course Instructors:

Dr. Sajid Muhaimin Choudhury
Assistant Professor

Ayan Biswas
Part Time Lecturer

Signature of Instructor: _____

Academic Honesty Statement:

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present),and cited all relevant sources while completing this project. We understand that if We fail to honor this agreement, we will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Anudwaipaon Antu** **Student ID: 1906001**	**Signature:** _____ **Full Name: Md. Sabbir Ahmed** **Student ID:1906004**
Signature: _____ **Full Name: Junayet Hossain** **Student ID: 1906026**	**Signature:** _____ **Full Name: Vivek Chowdhury** **Student ID:1906031**

Table of Contents

1 Abstract

This project explores all of the experiments covered in EEE 416 lab using STM32 Blue Pill instead of the STM Nucleo Board reducing the cost of the microcontroller significantly. The fundamental functions including GPIO, Timer, EXTI, ADC, SysTick are configured throughout the work. PCB with minimum floor area has been designed using Altium to illustrate the functionality in a compact form. This idea might be useful to replace the Nucleo board with a lot cheaper one for EEE 416 Laboratory.

2 Introduction

We use STM32F series microcontroller in our 'Microprocessor and Embedded System' Laboratory. F446 STM board used in laboratory currently, has a very wide range of functionality and comparatively large memory. However this advance specification comes with higher price. To perform the experiments of our lab, we do not require such advanced specifications, rather we can replace the F446 microcontroller with a much cheaper F103 series STM microcontroller. Our goal was to implement all the experiments using STM32F103C8T6 microcontroller, commonly known as Blue Pill.
To configure a new microcontroller, first we need to get familiar with the microcontroller, learn their functionality, get to know about the register configurations and finally able to apply them in experiments, also to find out the most suitable coding environment for the microcontroller.
All the proposed experiments can be performed using other microcontrollers like STM32 Black Pill, Arduino Uno etc.

3 Design

3.1 Problem Formulation (PO(b))

3.1.1 Identification of Scope

In our EEE 416 course, Microprocessor and Embedded System Laboratory, we did some basic experiments including basic GPIO configuration- turning LED on and off, configuring push button, running a stepper motor, configuring timer- LED Blink with timer, fading an LED, generating music using timer, configuring EXTI etc., also using SysTick interrupt, using sonar sensor, configuring DAC and ADC etc. on STM32F446RE Nucleo Board. We have conducted all the experiments on Keil uVision version 5.

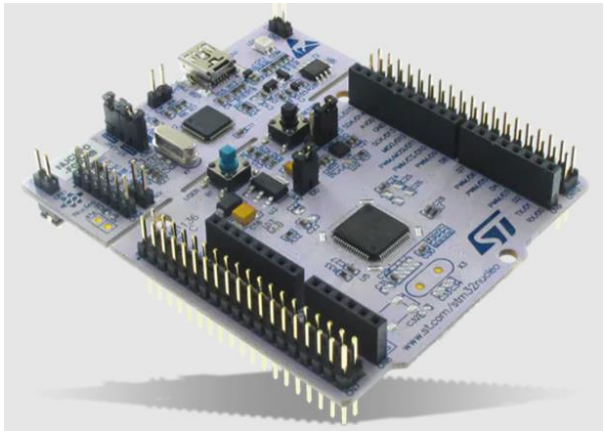


Figure: STM32F446RE Nucleo Board

Reducing Cost:

However, the cost of STM32F446RE Nucleo Board is around 4500 taka that is quite a lot. Recently many cheap STM32 boards are available in the market with almost the same functionalities of STM32F446RE Nucleo Board. One of them is STM32F103C8T6 or commonly known as STM32 Blue Pill. The astonishing fact is this STM32 Blue Pill costs only around 500 taka. Thus, an opportunity is created to replace the STM32F446RE Nucleo Board with the STM32 Blue Pill. This will greatly reduce costs.

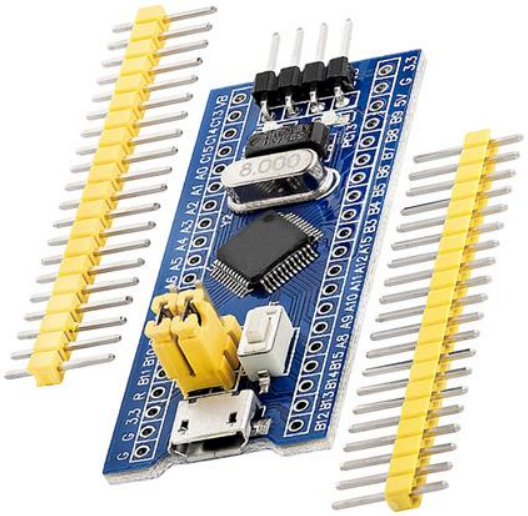


Figure: STM32F446RE Nucleo Board

Introducing another platform to program STM32 micro-controller:

As Keil uVision version 5 is being used in this laboratory, writing the base code every time or configuring a pin for different purposes is tiresome and often ends in a fatal error. STM32CubeIDE can be a better substitution for configuring the GPIO pin or creating an environment for specific purposes. STM32CubeIDE uses a Graphical User Interface to configure the pins. This is undoubtedly a great help in comparison with writing the base code for every operation.

3.1.2 Formulation of Problem

Performing the experiments in STM32 Blue Pill:

To replace the STM32F446RE Nucleo Board, we wanted to perform all the lab experiments in our EEE 416 Microprocessor and Embedded System Laboratory with STM32F103C8T6 micro controller board in bread board in the platform of STM32CubeIDE.

Designing a PCB:

Next, we planned to design a compact PCB where all the experiments can be conducted without any wire connection.

This would be a great leap in justifying the benefits or limitations of the STM32F103C8T6 Blue Pill Micro-controller whether it is possible to replace the STM32F446RE Nucleo Board with STM32F103C8T6 Blue Pill.

3.1.3 Analysis

Comparison between STM32F446RE Nucleo Board and STM32F103C8T6 Blue Pill:

Feature	STM32F446RE Nucleo Board	STM32F103C8T6 Blue Pill
Microcontroller	STM32F446RET6 (Cortex-M4 with FPU)	STM32F103C8T6 (Cortex-M3)
Clock Speed	Up to 180 MHz	Up to 72 MHz
Number of Timers	8	4
Number of ADC Channels	16	10
Number of DAC Channels	2 (12-bit)	None
Floating Point Unit (FPU)	Yes	No
USB, CAN, Ethernet	Yes	No
Flash Memory	512 KB	64 KB
RAM	128 KB	20 KB
ST-Link Debugger	Integrated on-board ST-Link debugger	External ST-Link debugger may be needed
Development Environment	STM32CubeIDE, STM32CubeMX, Keil uVision Version 5	STM32CubeIDE, STM32CubeMX Keil uVision Version 5
Cost	4500 Tk	500 Tk

In this comparison table, we clearly get that STM32F103C8T6 is relatively weaker than STM32F446RET6 Nucleo Board.

The experiments performed in the EEE 416 Lab:

- Exp-1: None
- Exp-2:
  - Interfacing Push Button
  - LED Blinking
  - Running a Stepper Motor
- Exp-3:
  - LED Blinking with Timer
  - LED Fading with Timer
  - Generating Music with Timer
- Exp-4:
  - Introducing Systick Delay
  - Driving SONAR Sensor
- Exp-5:
  - ADC
  - DAC

We mainly have to complete these experiments and design a PCB relating to these experiments. These experiments are mainly elementary level experiments to understand the basic functions of micro-controllers and learn to use them.

Despite being a weaker micro-controller in comparison with STM32F446RE Nucleo Board, STM32 Blue Pill has specifications good enough to run these experiments.

3.2 Design Method (PO(a))

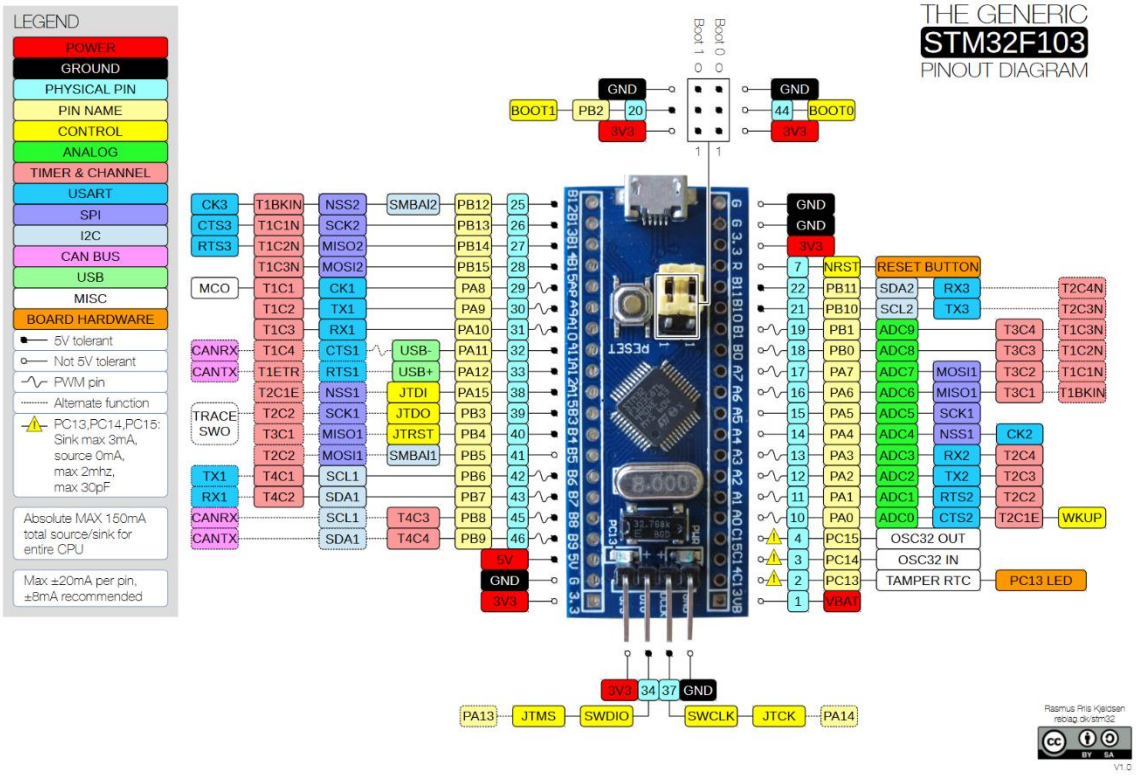
Micro controller familiarization: We had to develop a comprehensive understanding about the STM32 Blue Pill. We also had to study the microcontroller datasheet, reference manual and relevant documentation to understand the register orientation of the Blue Pill.

Peripheral Mapping and Relating with Nucleo Board: We had to identify the map existing peripherals and with STM32F103C8T6 with STM32F446RE.

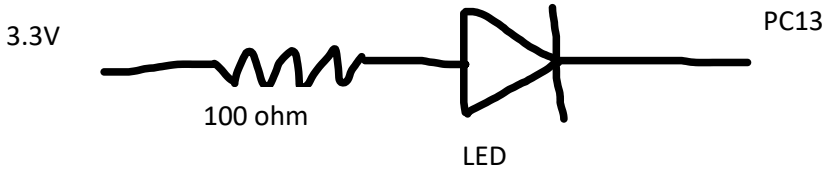
Using New Interface: We have used the STM32CubeIDE to program the blue pill. We had to get familiar with the whole new environment different from the lab and convert the program for STM32 Blue Pill.

3.3 Circuit Diagram

- Complete pin diagram of STM32F103C8T6:

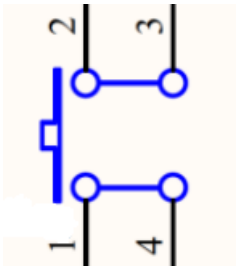


- Circuit diagrams for conducting experiment 2 in EEE-416 Lab:
  - Blinking an LED



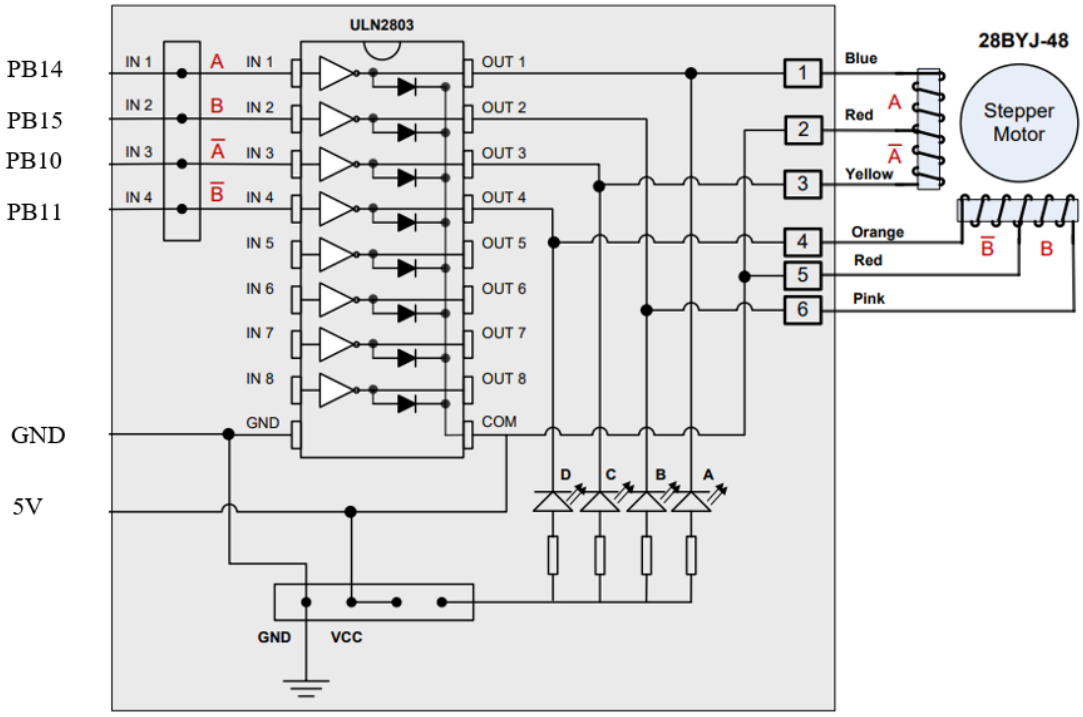
Anode of LED is connected to 3.3V of the STM board with a 100Ω series resistor and cathode is connected to PC13 pin.

- Toggle LED with Pushbutton



1 terminal of the push button is connected to 3.3V and terminal 2 is connected to PB13. Connection of LED is same as before.

- Run Stepper Motor in Full and Half Step

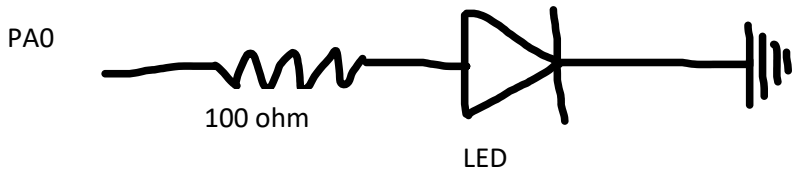


- Stepper motor reverse using push button

Stepper motor has same connection as before and push button is connected in same configuration as in toggling an LED.

➢ Circuit diagrams for conducting experiment 3:

- Blinking an LED with timer

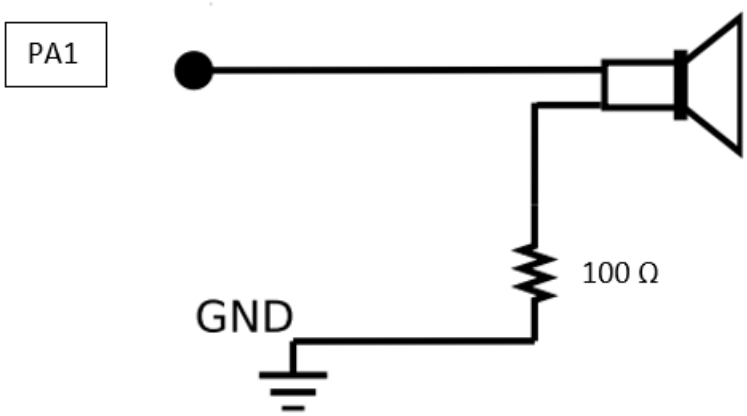


Anode of LED is connected to PA0 of the STM board with a 100Ω series resistor and cathode is connected to GND.

- Fading LED with PWM

Same LED connection as in blinking with timer.

- Music Generation with timer Output compare



Also, 5V is connected to supply power.

- LED Control with Push Button using External Interrupt

LED is connected between 3.3V and PC13 as in the part of 'Blinking LED' in experiment 2. Push button is connected between 3.3V and PB13 whose connection is same as 'Toggle LED with pushbutton' in experiment 2.

- Stepper motor reverse with push button using interrupt

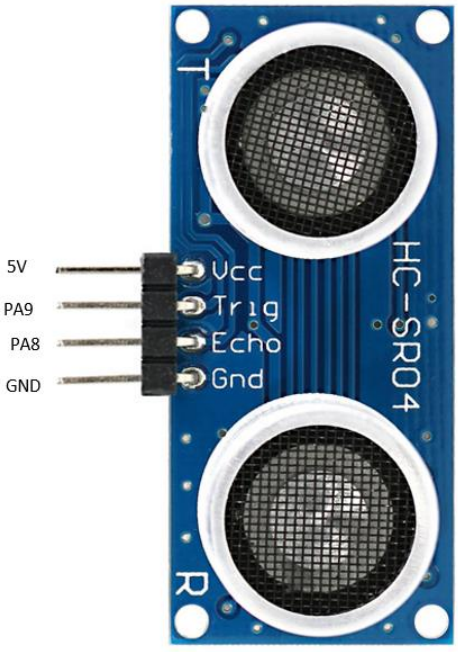
The same connection of the stepper as previously shown. A pushbutton configured as external interrupt is connected between 3.3V and PB13 as explained earlier.

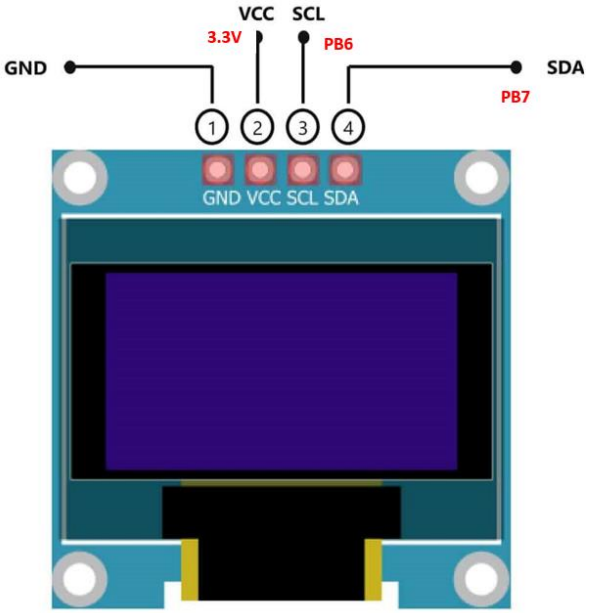
➢ Circuit diagrams for conducting experiment 4:

- LED Toggling using Systick

LED connected in same configuration as in 'Blinking LED' in experiment 2.

- Distance Measurement using Ultrasonic Sensor

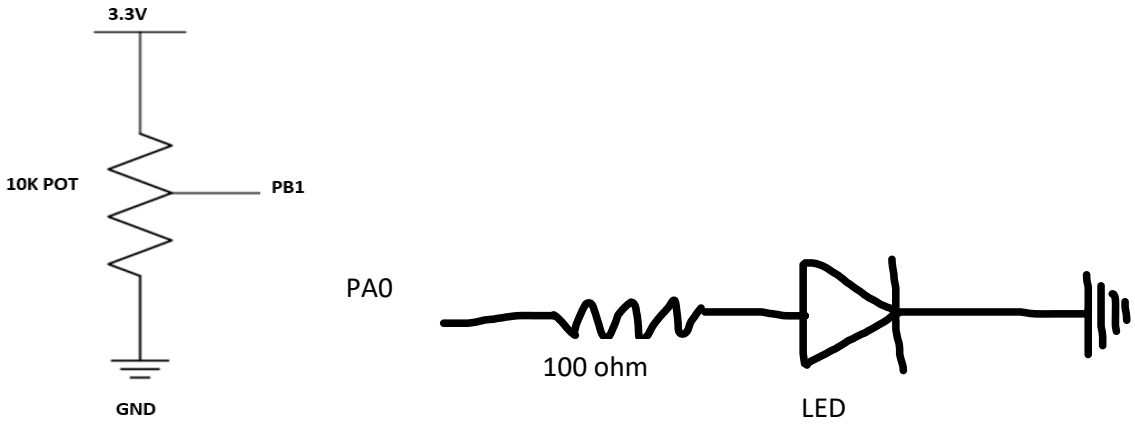




Trig and Echo pin of Ultrasonic sensor is connected to PA9 and PA8 pin respectively. For OLED display, SCL and SDA pins are connected to PB6 and PB7 pin respectively.

➢ Circuit diagrams for conducting experiment 5:

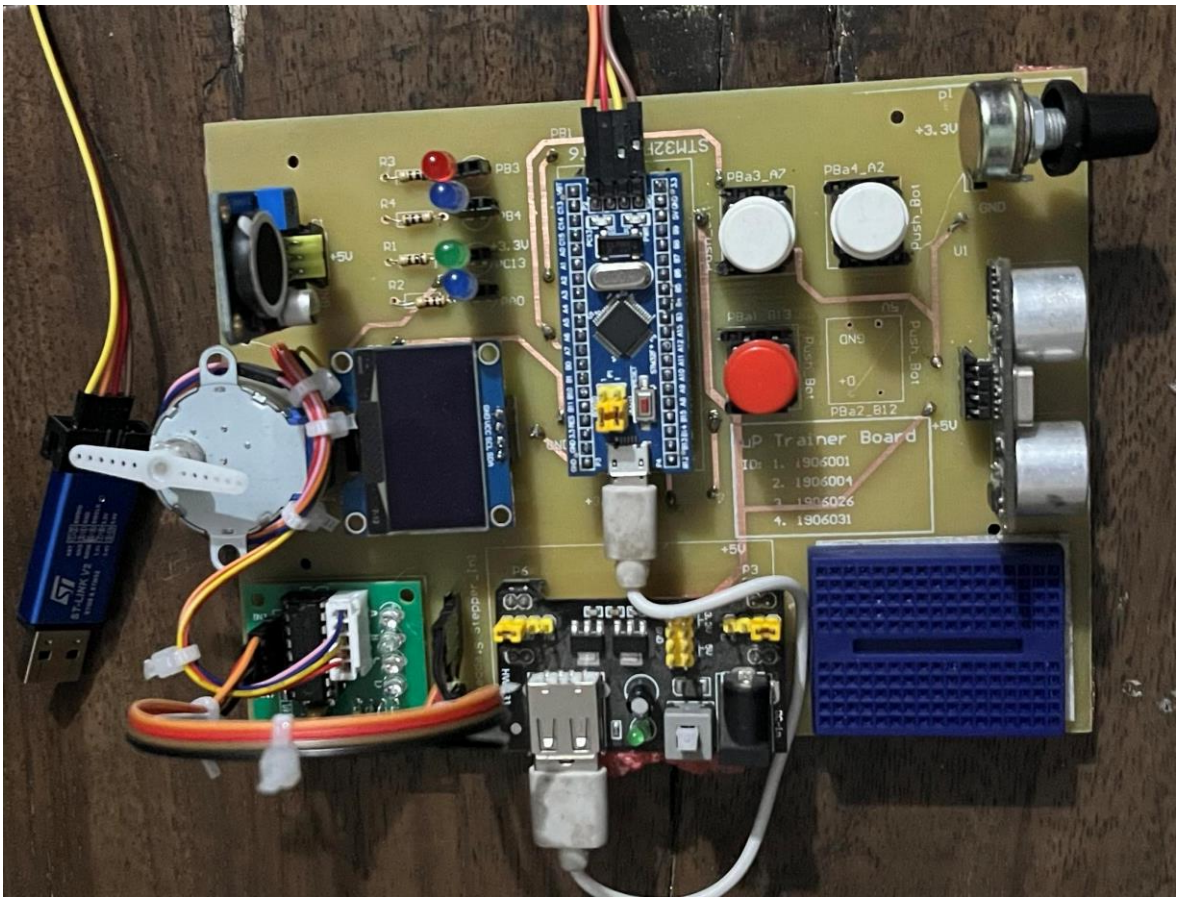
- Adjusting LED brightness using potentiometer and ADC



Potentiometer and LED are connected to the STM board as shown in the schematics.

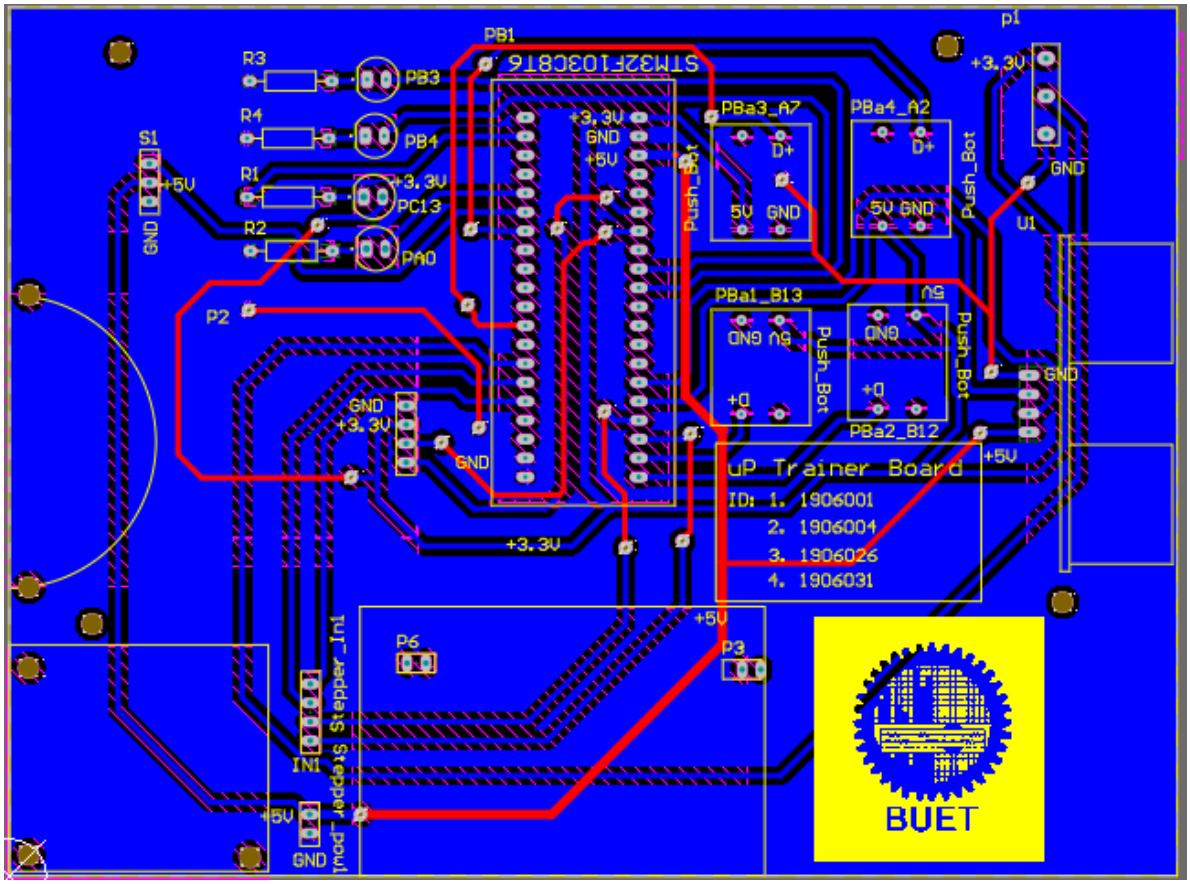
3.4 Hardware Design

Final structure after incorporating all the components together:



3.5 PCB Design

Design file:



3.6 Full Source Code of Firmware

Source code for ‘LED Control with Push Button using External Interrupt’ is attached. All the codes are given in the GitHub link in section 6.5

```
/* USER CODE BEGIN Header */
/**
* @file : main.c
.
*/
/* USER CODE END Header */
/* Includes ------------------------------------------
----------------------*/
#include "main.h"
/* Private includes ----------------------------------
----------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
/* Private typedef -----------------------------------
----------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
/* Private define ------------------------------------
----------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */
/* Private macro -------------------------------------
----------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */
/* Private variables ---------------------------------
----------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */
/* Private function prototypes -----------------------
----------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------
----------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
/**
* @brief The application entry point.
* @retval int
*/
uint8_t mode;
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
mode++;
if(mode>1) mode = 0 ;
}
int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration---------------------------------
--------------------*/

/* Reset of all peripherals, Initializes the Flash
interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */
/* Infinite loop */
```

```
/** Initializes the RCC Oscillators according to the
specified parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_0) != HAL_OK)
{
Error_Handler();
}
}

/**
* @brief GPIO Initialization Function
* @param None
* @retval None
*/
static void MX_GPIO_Init(void)
{
GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);

/*Configure GPIO pin : PC13 */
GPIO_InitStruct.Pin = GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : PB13 */
GPIO_InitStruct.Pin = GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 1, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
/**
* @brief This function is executed in case of error
occurrence.
* @retval None
*/
void Error_Handler(void)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the
HAL error return state */
__disable_irq();
```

```
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */
 if(mode == 0) {

          GPIOC->ODR |= (1<<13);

          HAL_Delay(50);
                                                }
                   if(mode == 1) {
         GPIOC->ODR &=~(1<<13);

          HAL_Delay(5);
                                                }
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

```
while (1)
{
}
/* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
/**
* @brief Reports the name of the source file and the
source line number
* where the assert_param error has occurred.
* @param file: pointer to the source file name
* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t *file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the
file name and line number,
ex: printf("Wrong parameters value: file %s on
line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

4 Implementation

4.1 Description

Initially, we designed a draft schematic for our PCB.

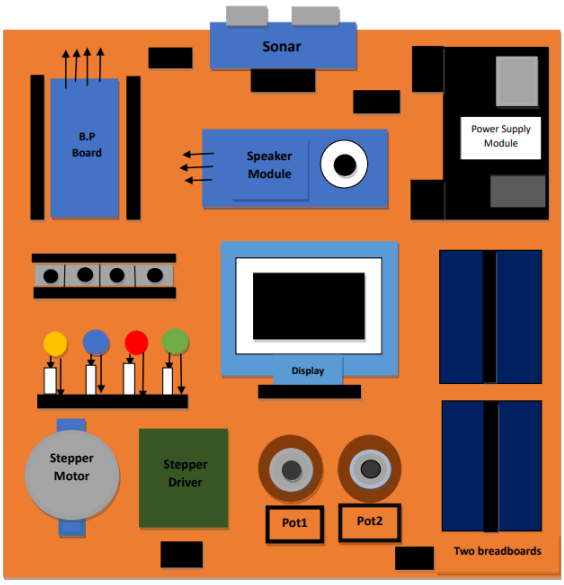


Figure 2: Draft PCB schematic of the trainer board

Here, we considered extra push buttons, LEDs and additional breadboards for the PCB. But as keeping the minimum PCB floor area was our priority, we have modified the design and built this new 3D model in Altium.

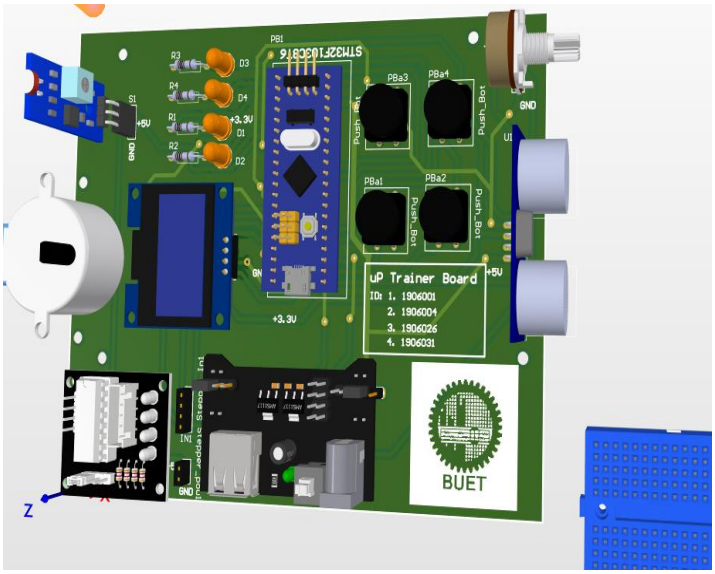


Figure 3: Modified PCB schematic (final)

The necessary schematics for the PCB of this model are shown below:

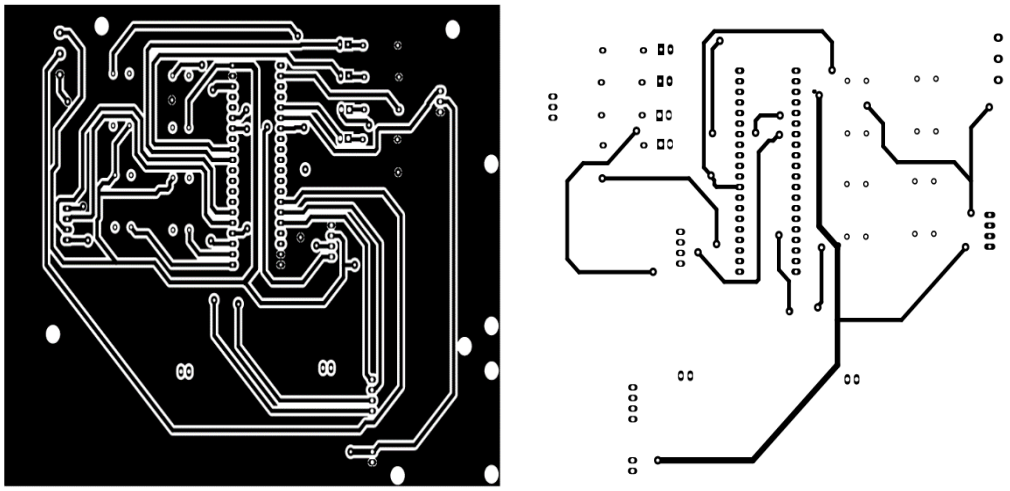


Figure 4: Bottom Copper (left) and top layer (right)

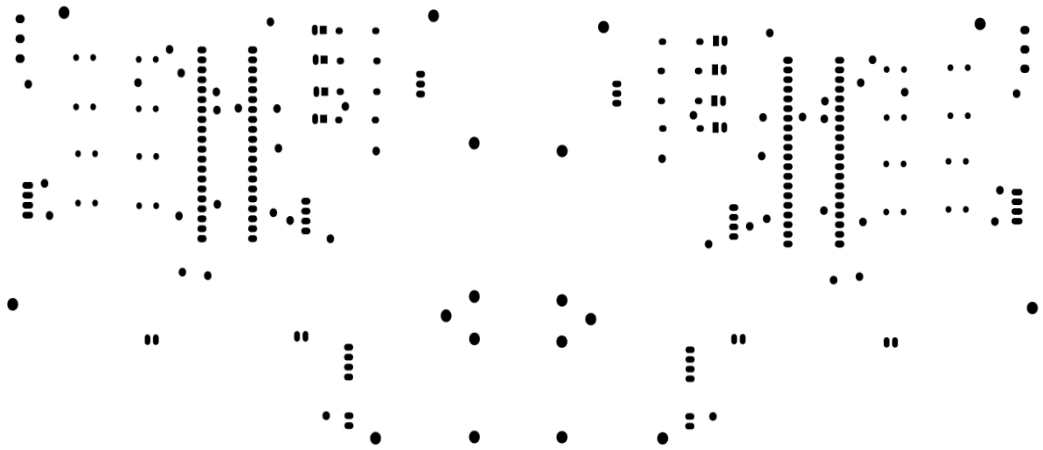


Figure 5: Bottom solder (left) and top solder (right)

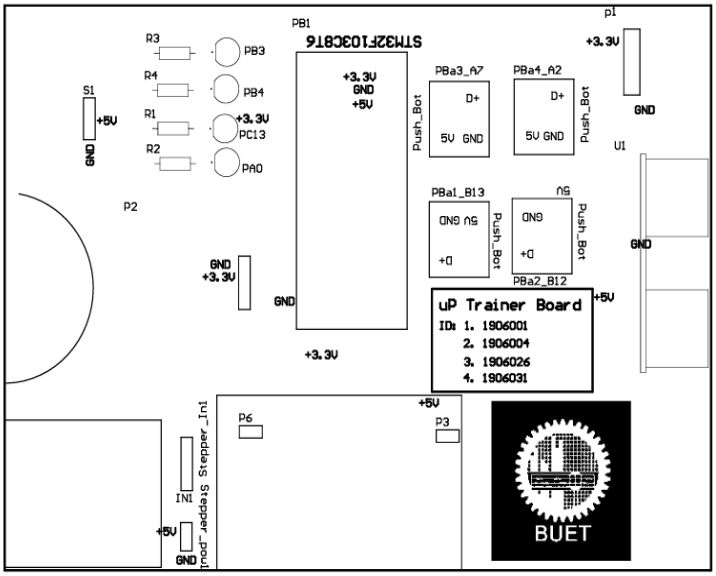


Figure 6: Top overlay

These are the necessary files for printing PCB. After finalizing this, we ordered two PCB boards from the service named Taru Projukty.

5 Design Analysis and Evaluation

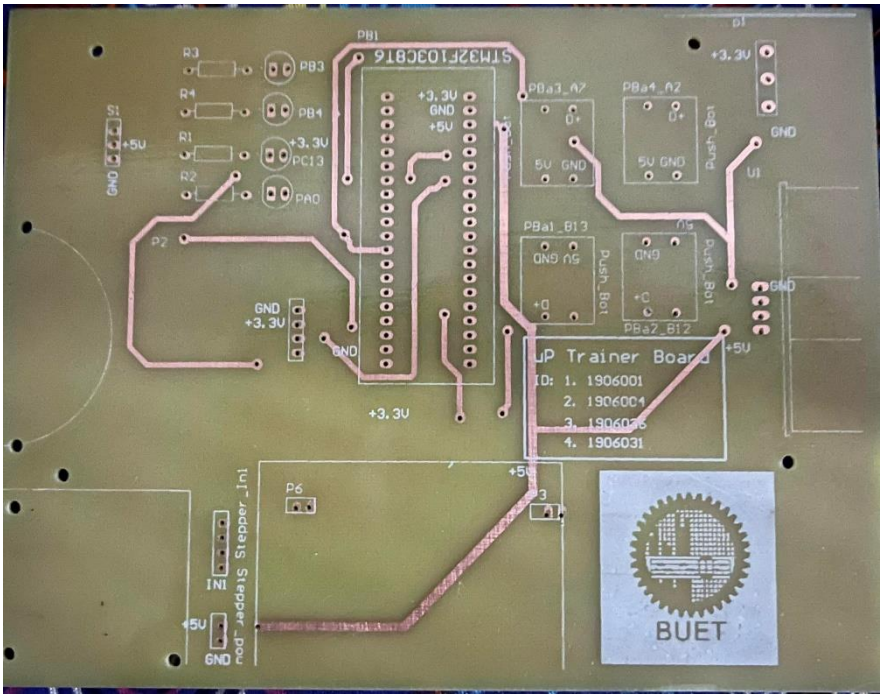


Figure 6: Front view of the PCB

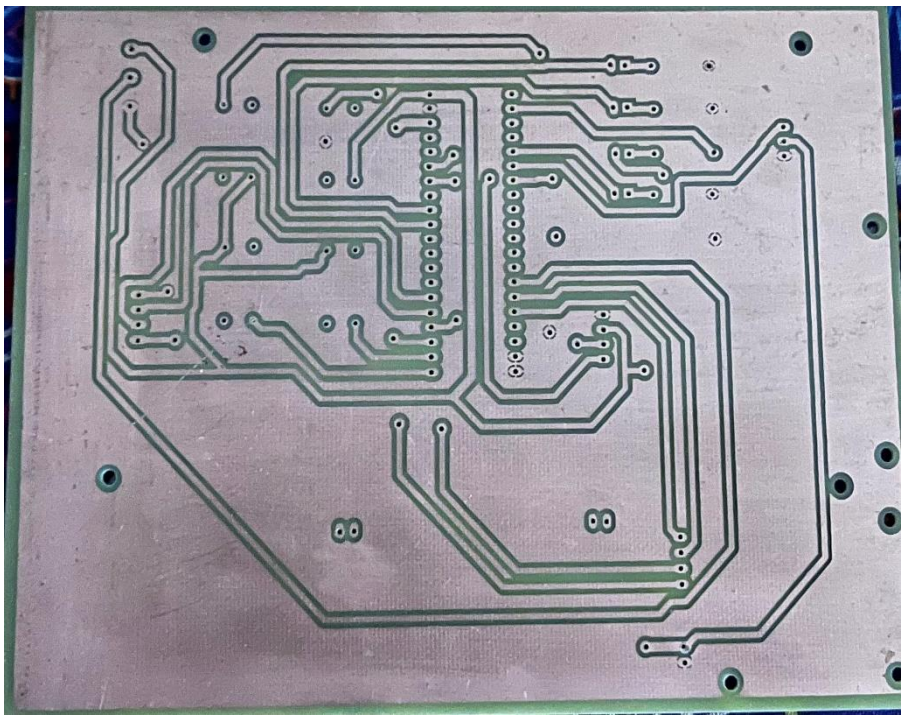


Figure 6: Front view of the PCB

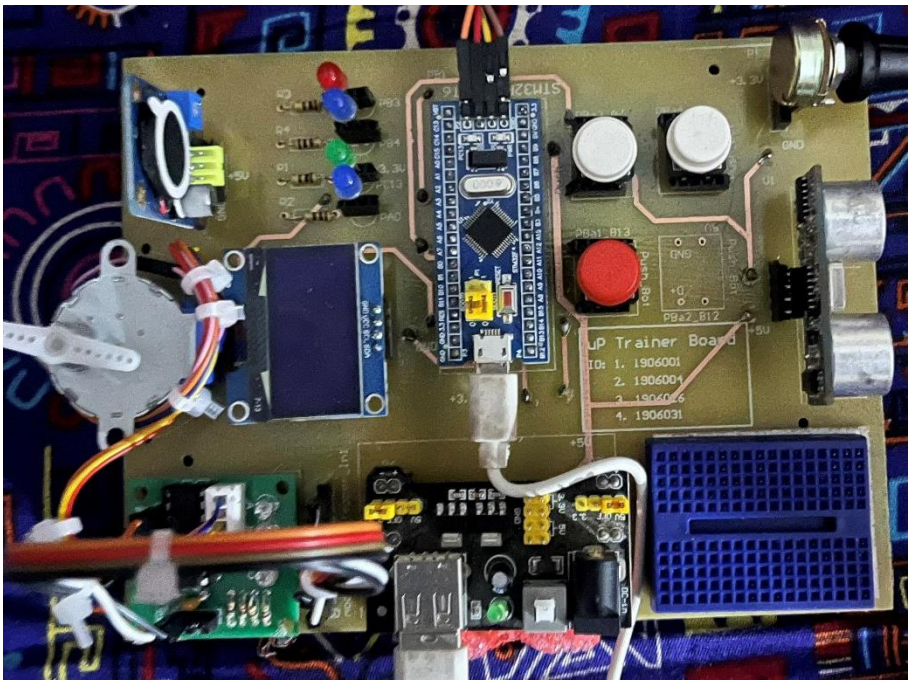


Figure 7: Front view after soldering all components

The quality of the PCB is average. We didn't have the masking because masking is done for bulk quantities of PCB. Besides, the diameter of the holes has not been maintained properly. That's why some of the components, like the stepper motor and the diver for stepper motor are mounted on the board via glue gun. Again, one of the holes for a push button has not been drilled. So, we did not solder it. Luckily, the default push button for the experiments is the red one. But the fourth push button can be added later if the hole has been drilled properly.

5.1 Novelty

The work of our project is novel because it is the perfect trainer board for performing all the experiments of newly designed EEE 416 and it is the first project of our batch to do so. Additionally, experiments and creative and innovative lab tests problems can also be performed here. The components of our trainer board are easily detachable. That's why defective components can easily be replaced without doing any harm to the trainer board. It can be a good device for teaching the basics of embedded systems to the students.

5.2 Design Considerations (PO(c))

For establishing the proper connection, we tried to use different pins for separate purpose. For example, we used PC13 for all the LED Blinking experiments configured as output (in MODER register). Likewise, we have used PA0 pin to show the results relating to timer and LED including LED blinking with timer and LED fading with timer.

While designing the PCB, we kept the track width 1mm as all the pins are signal pins (carries current in mA range). That's why we did not need thick track. Besides, we designed double layer PCB for keeping minimum floor area. The final size of our PCB is 16 cm (about 6.3 in) × 12 cm (about 4.72 in).

5.2.1 Considerations to public health and safety

In our PCB, we used a power supply module that can take both ac and dc and produce an output voltage of 5 V and 3.3 V. These low levels are safe for humans.

5.2.2 Considerations to environment

Made all the components detachable so that every PCB board can be reusable.

5.2.3 Considerations to cultural and societal needs

The manuals are well described for the trainer board. Users will find it easy to conduct the experiments. Besides, collaborative learning is also possible.

5.3 Limitations of Tools (PO(e))

The blue pill board does not have DAC (Digital to Analog Converter) module in it. So, the experiment of DAC cannot be performed here.

5.4 Impact Assessment (PO(f))

The trainer board is easier to use. Users will learn the concepts of embedded systems very easily.

5.4.1 Assessment of Health and Safety Issues

As the voltage levels are low and the currents are in mA range, no safety issues are present while using

the trainer board.

5.4.2 Assessment of Legal Issues

This project will be open source to teach embedded systems in a handy way.

5.3 Sustainability Evaluation (PO(g))

Education and Accessibility: Can be used for experiments relating to embedded systems. Because of its low cost and compact size, it will be easily accessible anywhere.

Open Source and Community Involvement: If the board is open source and encourages community involvement, it contributes positively to sustainability.

Longevity and Upgradability: A sustainable development board will be designed for longevity, allowing users to use it for an extended period without becoming obsolete quickly

Minimum Resource consumption: The dev board will use minimum resource in respect of hardware which is positive towards its sustainability.

- **Legal clearance:** As the resources are open source the production of the board will be legally ethical.

5.4 Ethical Issues (PO(h))

All the codes for the experiments have been configured by us. We also designed the PCB by our own. Any unethical means have been avoided completely.

6 Reflection on Individual and Teamwork (PO(i))

Every member was serious and helpful throughout the project. Tough times during the project include configuring all the base codes for the new microcontroller and debugging the PCB connection. All these problems have been solved with proper co-operation among the team members.

6.1 Individual Contribution of Each Member

1906001 & 1906004:

- Configuring and debugging the necessary codes for the experiments in blue pill using STM32CubeIDE.
- Hardware interfacing.

1906026 & 1906031:

- PCB designing with Altium.
- Soldering hardware components.

6.2 Mode of Teamwork

We divided our group into two parts. One group handled the software section and another group designed the hardware section. This reduced the workload and time.

6.3 Log-Book of Project Implementation

Date	Milestone achieved	Individual Role	Team Role	Comments
07.01.2024	Completed Exp-2	Made GPIO configuration	Helped to find resources	We worked as a team and completed all the tests, experiments
18.01.2024	Completed Exp-3	Figured out Timer and EXTI in CubeIDE	Gathered information about timer and interrupt	
26.01.2024	Completed Systick configuration	Configured SysTick in STM32CubeIDE	Finding resources	
01.02.2024	Completed configuring ADC	Configured ADC in ADC1 channel 9	Helped finding resources	
08.02.2024	Implemented the OLED display for SONAR Output	Found the library and configured the display	Helped to find the library for the display	
17.02.2024	Designing the PCB	Used Altium to design PCB	Helped in debugging and establishing connection	
24.02.2024	Placing all the all the components on the PCB	Soldering the PCB	Everyone checked the connections	
26.02.2024	Testing all the experiments on the PCB Board	Using STM32CubeIDE	Helped in hardware and software debugging	

6.4 Executive Summary

In this project, we have completed all the experiments of EEE 416 lab. Our purpose was to replace the STM32F446RE Nucleo Board with STM32F103C8T6 Blue pill board which cost only around 1/10 th of Nucleo Board. Though the Blue pill has weaker configuration in comparison with Nucleo board, the experiments in EEE 416 are basic experiments to understand the basic functions of the micro-controller like GPIO, Timer, EXTI, SysTick, ADC etc. Then we designed a PCB in Altium to demonstrate the experiments in a compact structure. Overall, this project may be a helpful one if we want to replace the Nucleo board with a much cheaper board for EEE 416 Laboratory.

6.5 Github Link

https://github.com/SabbirAhmedSaikot/Blue-Pill-STM-Dev-Board

6.6 YouTube Link

https://youtube.com/watch?v=8JwXmAnX6lI&si=W-Y7u9oS6dGKqnO7

7 Project Management and Cost Analysis (PO(k))

7.1 Bill of Materials

Component's Name	Cost
STM32 Blue pill Board (2 pcs)	1100
Potentiometer (10k, 100k, 50k) (2 pcs)	150
Stepper Motor with Driver	200
LED (20 pcs)	20
Sound Output Module	900
SONAR Sensor	100
Push Button (10 pcs)	80
OLED Display (2 pcs)	1100
PCB Board (2 pcs)	1160
Power Supply Module (2 pcs)	200
AC-DC Adapter	120
Male and Female Header	300
Soldering Iron+Lead	520
STLink V2 Debugger/Programmer	400
Total	6350 Tk

7.2Calculation of Per Unit Cost of Prototype

Component's Name	Cost
STM32 Blue pill Board	550
Push Button (4 pcs) (with cap)	50
LED (4 pcs)	5
Pot	30
OLED Display	550
Sound Output Module	900
SONAR	100
Power Supply Module	100
Resistor (100 ohm) (4 pcs)	5
Stepper Motor with Driver	200
Jumpers	10
Zip Tie	10
Male+Female Header	120
STLink V2	400
PCB	580
Total	3610 Tk

8 Future Work (PO(l))

- *Introducing Debouncing Circuit:* We have implemented software debouncing for the Push Buttons. However, hardware debouncing can be introduced using RC filter. It will increase the floor area of the PCB a bit larger but will surely improve the performance of the push buttons.

- *Introducing DAC:* The Blue Pill microcontroller has no internal system to perform DAC. It is possible to perform DAC using external circuitry.

- *Introducing USART:* In our lab experiments, we have not used USART communication. It is possible to add USART to perform more complex terms.

- *Getting Rid of External Power Supply:* We have used an external power supply to power up the blue pill externally. We are ensuring the proper voltage supply with smooth load and line regulation. However, it is possible to power up the blue pill with USB 2 port.

9 References

1. https://stm32-base.org/boards/STM32F103C8T6-Blue-Pill.html

2. https://erc-bpgc.github.io/handbook/electronics/Development_Boards/STM32/

3. https://predictabledesigns.com/introduction-stm32-blue-pill-stm32duino/

4. https://www.kevsrobots.com/resources/boards/stm32.html