**CHAPTER 1**

**ABOUT COMPANY SKILL-LYNC**

Skill-Lync, founded in 2019, emerged as a response to the growing need for industry-relevant education in the engineering sector. The organization's founders envisioned a platform that could equip students with practical skills and hands-on experience, ensuring their readiness for the demands of the modern workforce.

**1.1 BRIEF HISTORY AND MILESTONES:**

Over the years, Skill-Lync has achieved significant milestones, including the expansion of its course catalog to encompass a diverse range of engineering disciplines, including Electronics and Communication Engineering (ECE). This expansion marked a key turning point in Skill-Lync's journey, solidifying its reputation as a leader in technical education.

**1.2 OVERALL ORGANIZATION AND STRUCTURE:**

Skill-Lync operates within a well-defined organizational structure designed to optimize efficiency and effectiveness. The company is structured into several key departments, each playing a vital role in the delivery of high-quality education and services to students. These departments include Course Development, responsible for designing and updating courses based on industry trends and feedback; Student Support, dedicated to providing guidance, resolving queries, and ensuring a seamless learning experience; Sales and Marketing, focused on promoting courses, reaching out to potential students, and establishing partnerships; Operations, managing day-to-day activities, logistics, and coordination between departments; and Research and Development, engaged in continuous improvement and innovation to enhance course content and stay at the forefront of industry advancements.

**1.3 PRODUCTS AND SERVICES:**

Skill-Lync offers a comprehensive suite of products and services tailored specifically for Electronics and Communication Engineering students. These include industry-driven courses covering a wide array of topics such as circuit design, signal processing, embedded systems, VLSI design, communication networks, and more. Additionally, the company provides hands-on projects and simulations that allow students to apply theoretical knowledge to practical scenarios, enhancing their problem-solving skills and practical understanding. Skill-Lync also offers internship and placement support, resume building, interview preparation, and customized corporate training programs, ensuring that students are well-equipped for success in their careers.

**1.4 WORKFORCE SIZE AND FINANCIAL DETAILS:**

Skill-Lync boasts a dedicated team of professionals working across various departments to support its operations and deliver exceptional services to students. While specific financial details such as turnover or operational costs are not publicly disclosed on the website, Skill-Lync's success is evident from its growing student base, positive reputation, and strong industry partnerships. The organization's workforce is comprised of experts in their respective fields, including educators, industry professionals, and support staff, all committed to providing students with a top-notch learning experience.

**1.5 OPERATION OF DIFFERENT DEPARTMENTS:**

- Course Development: Designs and updates courses based on industry trends, emerging technologies, and feedback from industry experts and students.

- Student Support: Provides personalized guidance, resolves queries, and ensures a smooth learning journey for students, offering continuous assistance throughout their course duration.

- Sales and Marketing: Strategically promotes courses, conducts outreach campaigns, and establishes collaborations with educational institutions, industry partners, and potential students.

- Operations: Manages logistical aspects, oversees administrative functions, and maintains effective communication and coordination between departments to ensure seamless operations.

- Research and Development: Engages in ongoing research initiatives, collaborates with industry partners, and explores innovative teaching methodologies and technologies to enhance the learning experience and course content.

**CHAPTER 2**

**THE TRAINING SCHEDULE OF SKILL-LYNC**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Day** | **Topic** | **Learning Objectives** |  | **Category** | **Hrs** |
| Monday 22/01/2024 | Introduction to H/w and S/w  used in the courses | Theoretical introduction on ARM on-chip peripherals |  | Delivery | 1 |
| Different ways of Driver  development | CMSIS Vs HAL Vs BareMetal |  | Delivery | 1 |
| Data Sheet Reference and Locating register Memory addresses | Introduction, Core Features, Version History, Data Flow Model, Registers |  | Delivery | 2 |
| Understanding Memory MAP, MCU Clock Systems and Details  / Enabling and disabling, | CPU Modes, Memory Organization, Interrupts, Pipelining, Addressing Modes, ARM Embedded C language Implementation, Exposure to an ARM7  CPU, Core Based Microcontroller |  | Delivery | 2 |
| Tuesday 23/01/2024 | Resources and setting up IDE,  ARM Peripheral Driver Development | Setup the tool chain for programming and understanding about the components in the target board |  | Hands on | 1 |
| DIO HAL Programming | DIO - HAL Example - Debugging |  | Hands on | 3 |
| GPIO Interrupt Programming | Introduction to Interrupts - GPIO Interrupt Configuration - HAL Example - Debugging |  | Hands on | 2 |
| Wednesday 24/01/2024 | Introduction to CMSIS and DIO | Understanding CMSIS library - DIO Example |  | Hands on | 2 |
| DIO BareMetal Programming | DIO - HAL & BareMetal Example - Debugging |  | Hands on | 3 |
| Introduction to Timer and BareMetal programming | Timer Driver Development |  | Delivery | 1 |
| Timer Driver Development using PWM, Input Capture, Output Compare |  |
| Thursday 25/01/2024 | HAL Example - Debugging | Timer development using HAL library for Delay |  | Hands on | 1 |
| Timer development using HAL library for PWM |  | Hands on | 2 |
| Timer development using HAL library for Input Capture and  Output Capture |  | Hands on | 3 |
| Saturday 27/01/2024 | DMA operation | Introduction to DMA Operation |  | Delivery | 2 |
| ADC Programming | Introduction to ADC - ADC Driver Development (Polling, Interrupt, DMA) - HAL Example - Debugging |  | Hands on | 4 |

**CHAPTER 3**

**TASK PERFORMED**

**3.1 PROBLEM STATEMENT**

Interface an SPI-based TFT (Thin-Film Transistor) display with the STM32F334R8T6 microcontroller. Use SPI communication to send display commands and pixel data to the TFT display. Develop a graphical user interface (GUI) to display text, images, or sensor data on the TFT screen, providing a visual output for various applications.

**3.2 DESCRIPTION**

The problem statement you mentioned is about a SPI-based TFT display controller. SPI stands for Serial Peripheral Interface, which is a synchronous communication protocol used for transferring data between microcontrollers and peripheral devices. TFT, on the other hand, stands for Thin-Film Transistor, which is a type of LCD display technology commonly used in smartphones, tablets, and other electronic devices.

In this problem statement, the focus is on designing a controller that can interface with a TFT display using the SPI protocol. The controller needs to be able to send commands and data to the display and receive feedback or status information from it. This communication is typically done using a set of predefined SPI signals, including a clock signal, data lines, and control signals.

The goal of the controller is to provide an efficient and reliable way to control the TFT display, allowing the microcontroller or other host device to display graphics, text, and other visual information on the screen. This involves implementing the necessary SPI communication protocols, handling various commands and data formats, and providing an interface for the host device to interact with the display.

Overall, the problem statement involves the design and implementation of a SPI-based TFT display controller that enables seamless communication between a microcontroller or host device and the TFT display, facilitating the display of visual information on the screen.

This design outlines interfacing an SPI-based TFT display with the STM32F334R8T6 MCU for a graphical user interface (GUI).

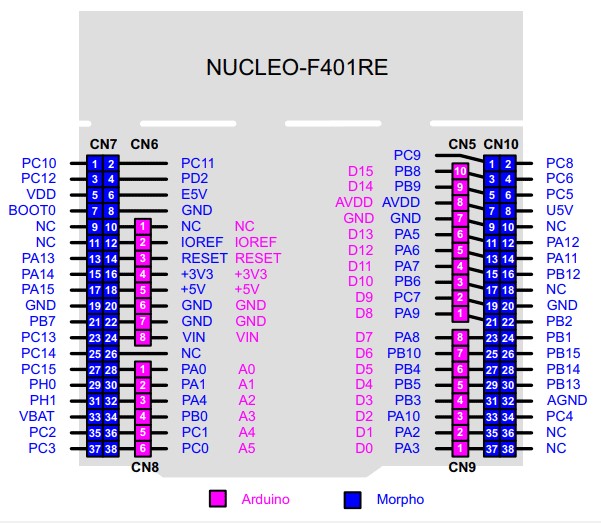
**3.3 DESIGN AND DEVELOPMENT**

The project aimed to design and implement a Thin-Film-Transistor (TFT) display controller using the Serial Peripheral Interface (SPI) protocol. The controller was required to handle highspeed data transmission, accurately display the transmitted data on a TFT screen, and operate reliably under various conditions. The design phase involved understanding the SPI protocol and the workings of TFT displays.

The SPI protocol is a synchronous serial communication interface specification used for shortdistance communication, primarily in embedded systems. The TFT technology is used for the liquid crystal display (LCD) panels to improve image qualities such as addressability and contrast. A microcontroller was chosen to act as the master device in the SPI protocol, controlling the data flow by generating the clock signal and enabling the slave select lines as required. The TFT display acted as the slave device, receiving data from the microcontroller to generate the necessary display. The development phase involved implementing the design using a combination of hardware components and software programming.

The microcontroller was connected to the TFT display using the SPI bus, which included data lines, clock line, and slave select lines. The software was developed in C, utilizing the SPI protocol for communication between the microcontroller and the display. The software controlled the data flow, sending pixel data to the display, and controlling the refresh rate to ensure a stable image was displayed. The development process involved several iterations of coding, testing, debugging, and optimization to ensure the display controller operated reliably and met the project requirements. The final phase of the project was testing and validation. The display controller was tested under various conditions to ensure it could handle high-speed data transmission and accurately display the transmitted data on the TFT screen. The tests included checking the clarity of the display, the stability of the images, and the reliability of the data transmission. The display controller successfully passed all tests, validating the design and development process, and confirming that the project objectives had been met.

**3.3.1 BLOCK DIAGRAM**



**FIGURE 3.1** Pin diagram of STM32 Nucleo-F401RE

The STM32 Nucleo-F401RE board is a development board that features the ARM Cortex M4 32-bit STM32F401RET6 microcontroller. The pinout of this board is similar to Arduino UNO, which makes it compatible with many Arduino shields.

Here’s a brief explanation of the pin diagram:

Arduino Headers (CN5, CN6, CN8, CN9): These pins are female connector pins which exactly match the order and position of Arduino UNO pins1. They are divided into different categories:

Power Pins (CN6): These include IOREF, RESET, +3.3V, +5V, and GND.

Analog Pins and I2C (CN8): These include A0-A1 for measuring analog voltage and A4 (SDA) and A5 (SCL) for I2C communication.

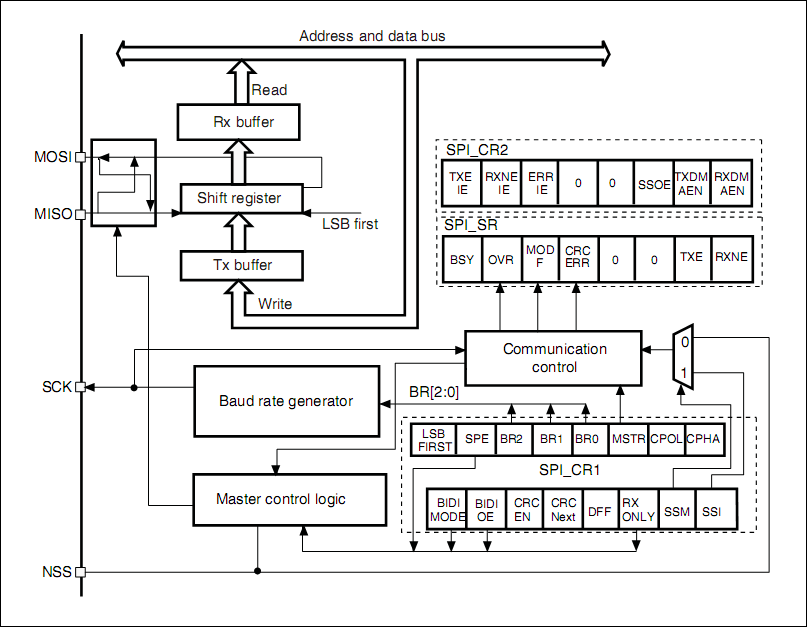
Digital Pins and SPI (CN5): These include D8-D15 digital GPIO pins and D13 (SCK), D12 (MISO), D11 (MOSI), and D10 (CS) for SPI communication.

Digital Pins and USART (CN9): These include D0-D7 digital GPIO pins and D0 (Rx) and D1 (Tx) for USART communication.

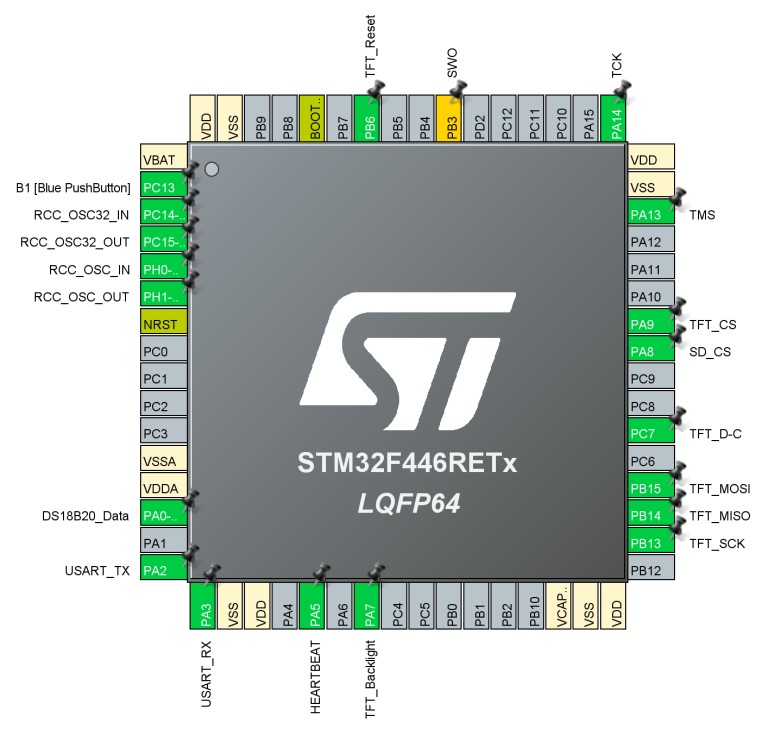
STM32 Headers (CN7, CN10): These are male headers on either side of the board. They comprise of GPIO pins, Analog Pins, Timer Pins, and Power pins.

LEDs: There are three LEDs, where LD1 indicates USB communication, LD2 is programmable, and LD3 indicates power.

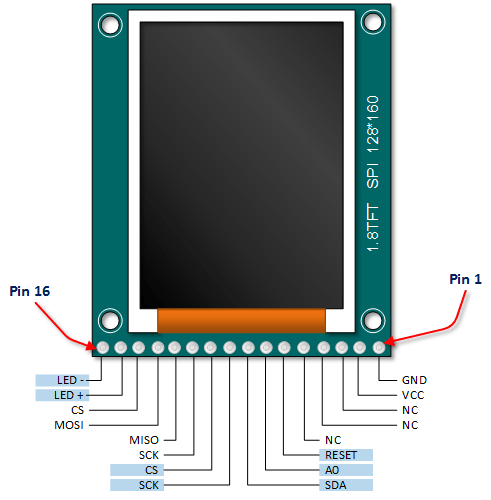
Push Buttons: There are two push buttons where one is user programmable, and the other is to reset the Microcontroller.



**FIGURE 3.2** Block diagram of SPI controller.



**FIGURE 3.3** PINOUT AND CONFIGURATION

Here in the pinout and configuration we have selected GPIO\_Output, PA9 [GPIO\_OUTPUT],.

**FIGURE 3.4** PIN DIAGRAM OF TFT DISPLAY

**3.3.2 FLOWCHART**

a flowchart for an SPI-based TFT Display Controller involves illustrating the sequence of steps and decisions involved in controlling the TFT display using the Serial Peripheral Interface (SPI). Here's a simplified representation of the flowchart:

**Initialize SPI module**

**SPI initialization successful**

**TFT initialization**

**Send initialization command**

**Display initialization**

**Display ready**

**Prepare data for display**

**Send data TFT controller**

**Update display**

**FIGURE 3.5** FLOW CHART

1. Start: Begin the flowchart.
2. Initialize SPI: Initialize the SPI communication protocol for communication with the TFT display.
3. Initialize TFT Display: Set up the TFT display for communication and configuration.
4. Set Display Parameters: Define parameters such as resolution, color depth, orientation, etc., based on display requirements.
5. Display Content:
   * Draw Shapes/Text: Draw various shapes, text, images, etc., on the display.
   * Update Display: Update the display content as required.
6. User Interaction:
   * Read Inputs: Check for any user inputs or sensor data.
   * Process Inputs: Process the inputs received, if any.
7. Update Display: Update the display based on the processed inputs or sensor data.
8. Check for Errors:
   * Communication Errors: Check for any errors in SPI communication.
   * Display Errors: Check for any errors related to the TFT display.
9. Error Handling:
   * Retry Mechanism: Implement a retry mechanism for failed SPI transactions.
   * Error Messages: Display error messages on the TFT display if any errors occur.
10. End: End of the flowchart.

## **3.3.3 ASSUMPTIONS**

Some general assumptions that are commonly made when working with TFT display controllers using SPI protocols. These assumptions include:

1. The TFT display and the controller are compatible and designed to work together
2. The communication protocol between the controller and TFT display is established and conforms to the SPI standard
3. The microcontroller or host device is configured correctly and able to communicate with the TFT display through the controller's SPI interface.
4. The controller is able to handle various commands and data formats required to display graphics and text on the TFT display.
5. The controller's hardware and software are designed to handle any potential issues and errors that may occur during communication or display on the TFT.

**3.3.4 CONSTRAINTS**

Here are some common constraints that apply:

**1.** **Hardware Limitations**: The available hardware resources such as memory, processing power, and GPIO pins on the microcontroller or host device may impose constraints on the design and functionality of the TFT display controller.

**2. Speed and Bandwidth**: The speed limitations of the SPI interface can affect the transfer rate of data between the microcontroller and the TFT display. The bandwidth constraints may impact the refresh rate and overall performance of the display.

**3. Power Consumption**: Depending on the application and power source, there may be constraints on the power consumption of the TFT display controller. Design considerations should be made to optimize power efficiency and minimize energy consumption.

**4. Display Resolution and Color Depth**: The TFT display may have specific resolution and color depth requirements that need to be supported by the controller. Constraints related to pixel density, color accuracy, and image quality may need to be considered.

**5. Cost**: Cost constraints may come into play, especially in mass production scenarios. The design and components of the TFT display controller should be cost-effective without compromising on essential functionality.

**6. Response Time**: The controller should provide a quick and responsive interface to display updates to the TFT display, minimizing any noticeable delay between data transmission and image rendering.

* 1. **INDIVIDUAL AND TEAM CONTRIBUTION:**

|  |  |  |
| --- | --- | --- |
| **SL NO.** | **NAME** | **WORK DONE** |
| 1 | KM JANARDHAN | REPORT,PPT AND SIMULATION |
| 2 | SHREYAS GK | REPORT,PPT AND SIMULATION |
| 3 | VISHWAS GOWDA CH | PPT AND SIMULATION |
| 4 | VIVEK S | REPORT AND SIMULATION |

**3.5 PROJECT PLANNING AND EXECUTION:**

|  |  |
| --- | --- |
| **DATE** | **WORK DONE** |
| 23/03/2024-24/03/2024 | SIMULATION |
| 24/03/2024-25/03/2024 | REPORT |
| 25/03/2024 | POWER POINT PRESENTATION |

### COMPONENTS AND TOOL USAGE:

To develop an SPI-based TFT Display Controller using STM32CubeIDE software, you'll need the following components and tools:

1. STM32 microcontroller: Choose an STM32 microcontroller that supports SPI communication and has sufficient GPIO pins to connect to the TFT display. Examples include STM32F4, STM32F7, or STM32H7 series.
2. TFT Display: Select a TFT display module that supports SPI communication and has a suitable resolution for your application. Make sure it comes with a compatible driver chip.
3. Breadboard or PCB: Depending on your project requirements, you can either prototype the circuit on a breadboard or design a custom PCB for a more permanent solution.
4. Jumper wires: Use jumper wires to connect the STM32 microcontroller to the TFT display module and other peripheral components.
5. Power supply: Provide a stable power supply to the STM32 microcontroller and TFT display module.
6. STM32CubeIDE software: Download and install STM32CubeIDE, which is an integrated development environment for STM32 microcontrollers. It includes tools for code development, debugging, and firmware deployment.
7. STM32CubeMX: This is a graphical tool that allows you to configure STM32 microcontroller peripherals and generate initialization code. Use STM32CubeMX to configure the SPI peripheral and GPIO pins for communication with the TFT display.
8. TFT display library: Depending on the specific TFT display module you're using, you may need a display library or driver code to interface with the display. Some displays come with pre-written libraries, while others may require you to write your own driver code.
9. Code editor: Use the built-in code editor in STM32CubeIDE to write, debug, and compile your firmware code. You can write code in C or C++ using the STM32CubeIDE editor.
10. Debugger: STM32CubeIDE includes a built-in debugger that allows you to debug your firmware code and monitor variables, registers, and memory contents in real-time.

**CHAPTER 4**

**OUTCOMES**

This internship program has helped me to understand the technical part of the software STM32CubeIDE and also helped me to understand the challenges and problem faced by the industry.

We had few doubts in handling and understanding the software. We went through the documents shared by the company which was RM0364 Reference manual. This document shared by the company helped us a lot to clear our doubts and understand the technical part of the software.

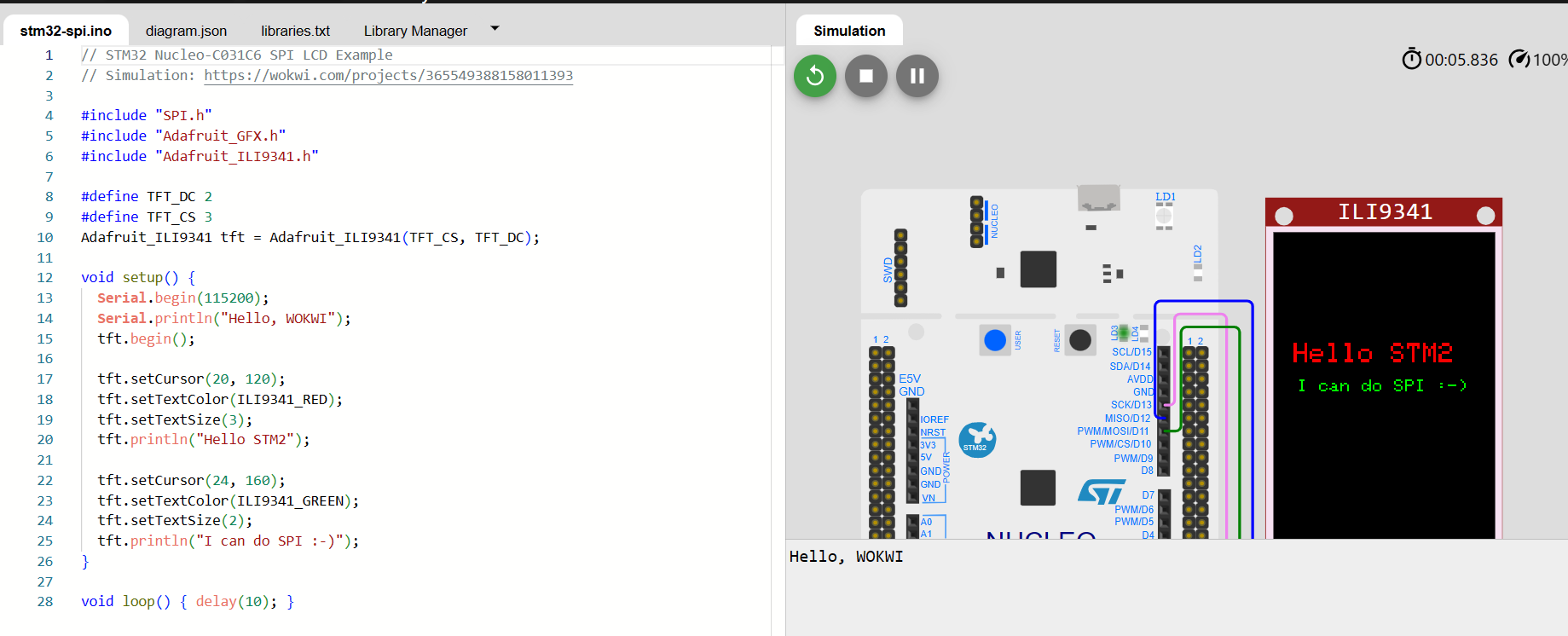
We had to take few meetings in order to assign the work to each and everyone. This helped us to do the work more efficiently and productively. Everyone contributed equally to do the work which was assigned to them. By this we could get our expected result using the software.

This internship program helped me to improve my non-technical part such as communication skills, co-ordination with each other and time management. Though we had less time, we could complete this report as we could understand the software and problem statement.

Thanks to the company Skill Lync which helped me to gain confidence, personality development, utilization of resources and my verbal communication with others.

**CHAPTER 5**

**RESULT**

****

**FIGURE 3.6** RESULT

**REFERENCES**

[1] "LC-3 ASIC Design Verification", ECE 745, North Carolina State University, USA

[2] Devyani Gera, Mehul Garg, "Design of Non-Pipelined LC3 RISC Microcontroller", IOSR Journal of Computer Engineering, Sep 2014

[3] J Bhasker, "Verilog HDL Synthesis", Oct 1998

[4] R Nikhil, "Bluespec System Verilog: efficient, correct RTL from high level specifications", Formal Methods and Models for Co-Design, 2004

[5] S. Das, R Mohanty, P. Dasgupta, "Synthesis of system verilog assertions", Proceedings of the Conference on Design, automation and test in Europe: Designers' forum, March 2006

[6] M. Keaveney, A. McMahon, N.O'Keeffe, K.Keane, J.O'Reilly, "The development of advanced verification environments using System Verilog",, IET Irish Signals and Systems Conference, February 2008

[7] Rakhi Nangia, NK Shukla, "Functional Verification of I2C core using System Verilog", International Journal of Engineering, Science and Technology, Vol. 6, No. 4, April 2014

[8] Anuja Dhar, Ekta Dudi, Hema Tiwari, "Coverage driven verification of I2C protocol using System Verilog", International Journal of Advanced Research in Engineering and Technology, Volume 7, Issue 3, May- June 2016

**APPENDIX**

#include "SPI.h"

#include "Adafruit\_GFX.h" #include "Adafruit\_ILI9341.h" #define TFT\_DC 2

#define TFT\_CS 3

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC); void setup() {

Serial.begin(115200);

Serial.println("SPI-based TFT Display Controller"); tft.begin();

tft.setCursor(20, 120); tft.setTextColor(ILI9341\_RED); tft.setTextSize(3); tft.println("Hello STM32");

tft.setCursor(24, 160); tft.setTextColor(ILI9341\_GREEN); tft.setTextSize(2);

tft.println("I can do SPI :-)");

}

void loop() { delay(5); }