**A PROJECT REPORT ON**

**Liquid Level Detection and Flow Control**

**REPORT SUBMITTED TOWARDS PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF**

**MASTER OF TECHNOLOGY**

**IN**

**Embedded Systems**

SUBMITTED BY

**GROUP NO:**

|  |  |
| --- | --- |
| **Name of the student**  **ANKITA GUPTA**  **ANIKETH MEHTA**  **SAILESH KUMAR PASAM** | **PRN**  **23070147001**  **23070147002**  **23070147005** |
|  |  |
|  |  |
|  |  |

****

**SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE – 412115**

**(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY))**

**A red logo with a globe and text

Description automatically generated**

**SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE – 412115**

**(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY))**

**2023-24**

**MASTER OF TECHNOLOGY**

**IN**

**Embedded Systems**

**CERTIFICATE**

This is to certify that **ANKITA GUPTA (23070147001), ANIKETH MEHTA (23070147002), AND SAILESH KUMAR PASAM (23070147005)**studying M.Tech in **EMBEDDED SYSTEM** have satisfactorily completed **LIQUID LEVEL DETECTION AND FLOW CONTROL** project inthe semester **I** during the academic year 2023-2024

**Signature of Course Instructor**

**ACKNOWLEDGMENT**

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project. First and foremost, we extend our deepest thanks to Dr. Priti Shahane, our project supervisor, for providing valuable guidance, support, and encouragement throughout the duration of this project. Your expertise and insights have been instrumental in shaping the direction of our work.

We are also grateful to the Department of E&TC of Symbiosis Institute of Technology for providing the necessary resources and facilities for the completion of this project. The conducive environment and access to relevant materials have greatly contributed to the quality of the outcome.

Lastly, I want to acknowledge the countless sources from various researchers, scholars, and authors whose work laid the foundation for our project. The wealth of knowledge available in the literature has been invaluable in shaping my understanding and approach.

This project would not have been possible without the collective support and contributions of all those mentioned above. Thank you for being a part of this journey.

ANKITA GUPTA - 23070147001

ANIKETH MEHTA - 23070147002

SAILESH KUMAR PASAM - 23070147005

**ABSTRACT**

Overflowing water is one of the most common and challenging issues that industries and households face. While water is essential to life, people tend to waste it by turning on the motor when their taps go dry and turning it off when the overhead tank overflows. This leads to unnecessary water wastage and sometimes even scarcity during times of crisis. Therefore, the proposed model's primary objective is to monitor the water levels in the tank and help manage it judiciously. To control liquid flow and detect parameters, we utilized an ARM controller and displayed data on a TFT screen.

**SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE – 412115**

**(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY))**

**2023-24**

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **LIST OF FIGURES** | | | i |
| **LIST OF TABLES** | | | ii |
| **LIST OF ABBREVIATION** | | | iii |
| **CHAPTER I: Introduction**  **CHAPTER II: Literature Survey**  **CHAPTER III: Experimental Work**  **CHAPTER IV: Result and Discussion**  **CHAPTER V: Conclusion and Future Work**  **References**  **Appendix** | | | 9  10  12  31  32  33  - |
|  | |
|  |  | |  |
|  | | |  |
|  |  | |  |
|  |  | |  |
|  | | |  |
|  |  | |  |
|  |  | |  |
|  | | |  |
|  | | |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig. No** | **Figure Name** | **Page No** |
| 1 | STM32F429ZIT6 | 13 |
| 2 | Four Channel Relay | 14 |
| 3 | Solenoid Valve | 14 |
| 4 | Ultrasonic Sensor | 15 |
| 5 | Flow Sensor | 16 |
| 6a | Hardware Implementation | 18 |
| 6b | Hardware Implementation | 18 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Name** | **Page No** |
| 1 | Ultrasonic Sensor Pin Configuration | 16 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviated Word** | **Expansion** |
| IoT | Internet of Things |
| WST | Water Storage Tank |
| ARM | Acorn RISC Machine |
| ADC | Analog to Digital Convertor |
| IDE | Integrated Development Environment |
| HAL | Hardware Abstraction Layer |
| UI | User Interface |
| GUI | Graphical User Interface |
| UART | Universal Asynchronous Receiver-Transmitter |

**CHAPTER I: INTRODUCTION**

The aim of this project is to design and implement a robust liquid-level detection and flow control system. The accurate monitoring and control of liquid levels are critical in various industrial and domestic applications such as chemical processing, water treatment, and irrigation. This project combines sensor technology, microcontroller programming, and actuator control to achieve precise liquid-level detection and flow regulation. We have proposed the ARM-based Liquid Level Detection & Flow Control System to automate the industrial task of detecting and monitoring liquid levels in containers. The system uses ultrasonic sensors, an ARM controller, and a motor to constantly monitor the liquid level. Once the level reaches the required limit, the system turns off the motor, saving energy. This system eliminates the need for constant monitoring, making it a useful and time-saving tool for industries.

**CHAPTER II: LITERATURE SURVEY**

* S Gunde et al. have proposed [1] the system for monitoring automatic water flow using IoT is a state-of-the-art solution that allows for the automatic control of submersible pumps based on water levels. The system uses an ultrasonic sensor to detect the water level, which triggers the submersible pump to start automatically when the tank is empty. Once the water level reaches the maximum point, the submersible pump stops. In addition, an SMS notification is sent to the user when the water level reaches the maximum point. The system also displays the water levels in both percentages and centimeters on a graph, which is used to monitor the levels. All the water level units are stored in the cloud, providing an easy and convenient way to access the data from anywhere. Overall, this system is an efficient and reliable solution that ensures the water level is maintained at an optimum level, providing an uninterrupted water supply while conserving water resources. It is a great example of how IoT can be used to solve everyday problems.
* Farmanullah Jan et al. [2] have concluded various methods in their literature survey for smart water tanks. Access to freshwater is a basic human right, yet a significant percentage of the global population lacks access to it for domestic uses. Over urbanization, industrial growth, and global warming are depleting water resources in many countries, making it a pressing issue. In developing countries, water is stored in aboveground or underground tanks, which are manually maintained, leading to water wastage due to human negligence. Additionally, decaying infrastructure can cause water leakage from storage tanks and supply pipes. To address these issues, researchers are turning to IoT technology to efficiently monitor water levels, detect leakage, and automatically refill tanks whenever required. This technology also provides real-time feedback to users and experts through a webpage or smartphone. However, there are few review articles on IoT-based solutions to monitor water levels, detect water leakage, and control water pumps at the individual level. To fill this gap in the literature, this study provides a review of IoT-controlled water storage tanks (IoT-WST), detailing current techniques and technologies in IoT-WST, selection of appropriate hardware, and securing an IoT cloud server.
* Kiran S. Shingote and Priti Shahane [3] propose a microcontroller-based design for a flow control system that automates the gate in canal operations. The proposed system aims to improve irrigation operational efficiency, power use, accuracy in measurement, water distribution, and reactions to imbalance. The system consists of several sub-systems, including the RTU, solar power system, flow measurement system, level measurement system, gate actuator system, and communication system. The Remote Terminal Unit (RTU) monitors upstream level, downstream level, downstream flow, power status, gate opening, gate health, and security. The microcontroller-based flow control system is an improved design compared to PLC and is cost-effective. It can achieve all the needs with continuous monitoring and control of the flow to water users. The system's response time is very fast, and it sends immediate alarms in case of vandalism to the main SCADA. Therefore, reacting to imbalances can be achieved. This system can reduce water wastage and labor dependency, and it does not require human intervention, reducing the manpower involved in canal operation. The proposed system can be successfully implemented for the smooth operation of the canal system, considering human disturbances. Hence, the microcontroller-based flow control system is an attractive solution to replace the existing conventional, as well as PLC operated flow control system for irrigation canal automation.

**CHAPTER III: EXPERIMENTAL WORK**

* **Methodology**

The proposed system is designed to monitor water flow automatically using IoT technology. It utilizes an ultrasonic sensor to detect the water level, and based on the readings, the submersible pump starts automatically when the tank is empty. As the water level rises to the maximum point, the submersible pump automatically stops. The water levels are displayed in the form of a graph and the values are shown in both percentage and centimeter units. These water level readings are stored securely in the cloud.

In this project, we will be using the STM32F429ZIT6 Board to interface various sensors and control the flow of liquid. The first step is to select the appropriate board, which provides us with the necessary GPIO pins to connect the sensors.

Next, we will interface the sensors with the Microcontroller Board by connecting them to GPIO pins. This will enable us to receive the readings from the sensors and process them using the board.

To convert the analog sensor readings into digital values, we will implement an Analog-to-Digital Converter (ADC). This will allow us to process the sensor data and obtain accurate readings.

Using Liquid level sensors, we can determine the height of the liquid. Additionally, we can calculate the flow rate of the liquid using a flow sensor input. By setting a liquid threshold, we can monitor the water level. If the water level exceeds the upper threshold level, the TFT screen will display red colour. Similarly, if the water level is less than the lower threshold, yellow colour will be displayed on the TFT screen.

In case the liquid flow rate is more than the desired value, detected by a pressure sensor, we can regulate the flow of liquid by interfacing a solenoid valve. This will ensure that the liquid flow rate is maintained at the desired level.

This project is a great example of how we can use a Microcontroller Board to interface with various sensors and control the flow of liquid. It can be used in a variety of applications, such as in industrial settings or in household appliances.

**Hardware Components Used:**

* **STM32F429ZIT6 -**

Fig:1**A close-up of a green electronic device

Description automatically generated**The STM32F429ZIT6 is a microcontroller from STMicroelectronics' STM32 family, based on the ARM Cortex-M architecture. It belongs to the high-performance F4 series and includes various peripherals and features. The name STM32F429ZIT6 is broken down as follows: STM32 represents the brand name for a family of 32-bit microcontrollers; F4 series is one of the series within the STM32 family, known for its high-performance capabilities; 29 signifies that it belongs to a particular generation within the F4 series; ZI denotes specific features or characteristics of the microcontroller, which can be found in the datasheet; and T6 indicates specific package and temperature range options. For more detailed information, the datasheet and reference manual provided by STMicroelectronics are recommended**.**

* **Four Channel Relay -**

Fig:2**A blue circuit board with several blue switches

Description automatically generated**A four-channel relay module is a device that has four individual relays on a single board, each of which can be controlled independently. This allows you to control four different electrical circuits or devices independently, turning them on or off as needed. The module is designed to be controlled by external signals, typically digital signals from a microcontroller or a similar device. Relay modules often include optocouplers or other means of electrical isolation between the control circuit and the switched circuit. They are commonly used in home automation, robotics, industrial control systems, and other applications where multiple electrical devices need to be controlled independently. Always check the electrical specifications and take proper safety precautions when using a four-channel relay module.

* **Solenoid Valve –**

Fig:3**A close-up of a valve

Description automatically generated**A solenoid valve is an electromechanical device that regulates liquid or gas flow in a system. It comprises a solenoid and a valve. The solenoid creates a magnetic field that moves the plunger to open or close the valve, allowing or blocking fluid flow. Solenoid valves are widely used in various systems, from HVAC to irrigation, for remotely controlling fluid flow.

* **Ultrasonic Sensor –**

Fig:4**A blue circuit board with two round speakers

Description automatically generated**Ultrasonic sensors use ultrasonic sound waves to measure distance, detect objects, and navigate in various applications. They emit ultrasonic waves and measure the time it takes for them to reflect off an object and return to the sensor, which is then used to calculate the distance between the sensor and the object. Ultrasonic sensors are widely used in robotics, industrial automation, automotive parking systems, and security systems.

**Ultrasonic Sensor Pin Configuration**

|  |  |  |
| --- | --- | --- |
| **PIN NUMBER** | **PIN NAME** | **DESCRIPTION** |
| 1 | VCC | THE VCC PIN POWERS THE SENSOR |
| 2 | TRIGGER | TRIGGER PIN IS AN INPUT PIN |
| 3 | ECHO | ECHO PIN IS AN OUTPUT PIN |
| 4 | GROUND | THIS PIN IS CONNECTED TO THE GROUND OF THE SYSTEM |

Table:1

* **Flow Sensor –**

Fig:5**A black plastic pipe with wires

Description automatically generated with medium confidence**Flow sensors measure fluid flow and are crucial in various industrial, commercial, and residential applications. They provide real-time information about flow rate, which is valuable for process control, system optimization, and ensuring proper equipment functioning. Different types of flow sensors are used for specific applications and require calibration for accurate measurements**.**

**Software Components Used**

* **STM32 CUBE MX**

STM32CubeMX is a graphical software configuration tool by STMicroelectronics that assists developers in configuring STM32 microcontroller projects and generating initialization code. It offers a user-friendly interface, supports a wide range of STM32 microcontrollers, and allows developers to configure peripherals like GPIO, USART, SPI, I2C, and timers. It also supports clock configuration, pinout configuration, middleware and RTOS integration, power consumption analysis, peripheral initialization code generation, code generation options, hardware abstraction layer (HAL) configuration, and firmware updates. STM32CubeMX is closely associated with the STM32Cube HAL, providing consistent APIs for accessing microcontroller peripherals. It also supports various programming languages and supports the latest microcontrollers and features. STM32CubeMX is a versatile tool for both beginners and experienced developers.

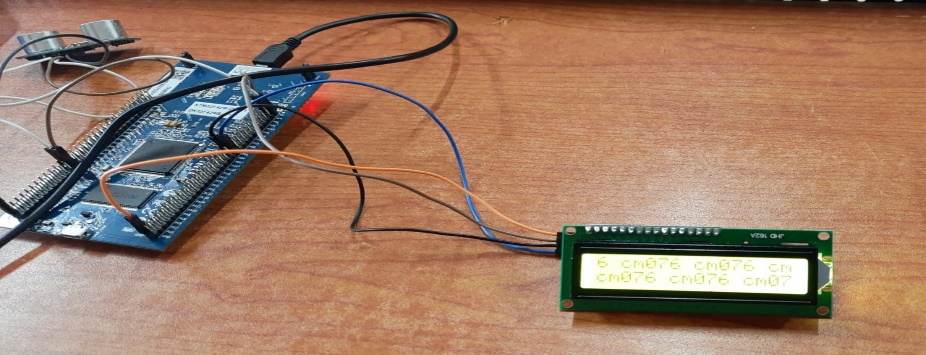
* **STM32 CUBE IDE**

STM32CubeIDE is an integrated development environment (IDE) by STMicroelectronics for software development for STM32 microcontrollers. It is built on the Eclipse framework, allowing users to benefit from its extensibility and plugin ecosystem. STM32CubeIDE supports a wide range of STM32 microcontrollers, including STM32CubeMX, and includes a code editor for organizing projects and files. It also integrates with STM32CubeMX, allowing users to configure microcontroller peripherals graphically and generate initialization code. The IDE also provides debugging tools, including a debugger and trace features, and supports real-time operating systems like FreeRTOS. Developers can choose programming languages and generate code in various formats, and it supports automatic project build and compilation. STM32CubeIDE also supports collaborative development through version control systems like Git, enabling teams to work together on STM32 projects.

* **touchGFX**

TouchGFX is a user interface (UI) development framework designed for embedded systems, specifically targeting microcontrollers with graphical user interfaces (GUIs). It simplifies the process of designing and implementing visually appealing and responsive touch-based interfaces for devices like microcontroller-powered displays. TouchGFX is closely integrated with STM32 microcontrollers, leveraging their hardware graphics acceleration features for efficient UI rendering. It is optimized for touch-based interaction, allowing developers to create touch-sensitive interfaces for displays, such as industrial control panels and home automation interfaces. The framework includes tools for designing graphics and animations, a WYSIWYG editor, optimized memory usage, code generation, performance optimization, customizable themes, and community support. It is crucial for embedded systems with limited resources and is supported by an active community of developers and users.

**Hardware Implementation**

**** fig:6a

**A circuit board with wires

Description automatically generated** fig:6b

**Code**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "i2c-lcd.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 ---------------------------------------------------------\*/

I2C\_HandleTypeDef hi2c1;

TIM\_HandleTypeDef htim1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_I2C1\_Init(void);

static void MX\_TIM1\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

void delay (uint16\_t time)

{

\_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0);

while (\_\_HAL\_TIM\_GET\_COUNTER (&htim1) < time);

}

uint32\_t IC\_Val1 = 0;

uint32\_t IC\_Val2 = 0;

uint32\_t Difference = 0;

uint8\_t Is\_First\_Captured = 0; // is the first value captured ?

uint8\_t Distance = 0;

#define TRIG\_PIN GPIO\_PIN\_8

#define TRIG\_PORT GPIOE

// Let's write the callback function

void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)

{

if (htim->Channel == HAL\_TIM\_ACTIVE\_CHANNEL\_1) // if the interrupt source is channel1

{

if (Is\_First\_Captured==0) // if the first value is not captured

{

IC\_Val1 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); // read the first value

Is\_First\_Captured = 1; // set the first captured as true

// Now change the polarity to falling edge

\_\_HAL\_TIM\_SET\_CAPTUREPOLARITY(htim, TIM\_CHANNEL\_1, TIM\_INPUTCHANNELPOLARITY\_FALLING);

}

else if (Is\_First\_Captured==1) // if the first is already captured

{

IC\_Val2 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); // read second value

\_\_HAL\_TIM\_SET\_COUNTER(htim, 0); // reset the counter

if (IC\_Val2 > IC\_Val1)

{

Difference = IC\_Val2-IC\_Val1;

}

else if (IC\_Val1 > IC\_Val2)

{

Difference = (0xffff - IC\_Val1) + IC\_Val2;

}

Distance = Difference \* .034/2;

Is\_First\_Captured = 0; // set it back to false

// set polarity to rising edge

\_\_HAL\_TIM\_SET\_CAPTUREPOLARITY(htim, TIM\_CHANNEL\_1, TIM\_INPUTCHANNELPOLARITY\_RISING);

\_\_HAL\_TIM\_DISABLE\_IT(&htim1, TIM\_IT\_CC1);

}

}

}

void HCSR04\_Read (void)

{

HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, GPIO\_PIN\_SET); // pull the TRIG pin HIGH

delay(10); // wait for 10 us

HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, GPIO\_PIN\_RESET); // pull the TRIG pin low

\_\_HAL\_TIM\_ENABLE\_IT(&htim1, TIM\_IT\_CC1);

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

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();

MX\_I2C1\_Init();

MX\_TIM1\_Init();

/\* USER CODE BEGIN 2 \*/

lcd\_init();

HAL\_TIM\_IC\_Start\_IT(&htim1, TIM\_CHANNEL\_1);

lcd\_send\_string ("Dist= ");

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

HCSR04\_Read();

lcd\_send\_data((Distance/100) + 48); // 100th pos

lcd\_send\_data(((Distance/10)%10) +48); // 10th pos

lcd\_send\_data((Distance%10)+48); // 1st pos

lcd\_send\_string(" cm");

HAL\_Delay(200);

/\* 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};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE3);

/\*\* 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 I2C1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C1\_Init(void)

{

/\* USER CODE BEGIN I2C1\_Init 0 \*/

/\* USER CODE END I2C1\_Init 0 \*/

/\* USER CODE BEGIN I2C1\_Init 1 \*/

/\* USER CODE END I2C1\_Init 1 \*/

hi2c1.Instance = I2C1;

hi2c1.Init.ClockSpeed = 100000;

hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Analogue filter

\*/

if (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c1, I2C\_ANALOGFILTER\_ENABLE) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Digital filter

\*/

if (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c1, 0) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C1\_Init 2 \*/

/\* USER CODE END I2C1\_Init 2 \*/

}

/\*\*

\* @brief TIM1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM1\_Init(void)

{

/\* USER CODE BEGIN TIM1\_Init 0 \*/

/\* USER CODE END TIM1\_Init 0 \*/

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_IC\_InitTypeDef sConfigIC = {0};

/\* USER CODE BEGIN TIM1\_Init 1 \*/

/\* USER CODE END TIM1\_Init 1 \*/

htim1.Instance = TIM1;

htim1.Init.Prescaler = 72-1;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 0xffff-1;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_IC\_Init(&htim1) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigIC.ICPolarity = TIM\_INPUTCHANNELPOLARITY\_RISING;

sConfigIC.ICSelection = TIM\_ICSELECTION\_DIRECTTI;

sConfigIC.ICPrescaler = TIM\_ICPSC\_DIV1;

sConfigIC.ICFilter = 0;

if (HAL\_TIM\_IC\_ConfigChannel(&htim1, &sConfigIC, TIM\_CHANNEL\_1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM1\_Init 2 \*/

/\* USER CODE END TIM1\_Init 2 \*/

}

/\*\*

\* @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\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOE, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : PE8 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_8;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\* 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();

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 \*/

**CHAPTER IV: RESULT AND DISCUSSION**

A project focusing on liquid-level detection using an ultrasonic sensor and communication of the measured distance via UART for monitoring purposes. The key components include an ultrasonic sensor, an ADC (Analog-to-Digital Converter), and a UART module. The main functionality of the code involves continuously measuring the distance from the ultrasonic sensor using ADC, converting the ADC value to a distance, and then transmitting this distance information over UART. The code includes an ultrasonic sensor for distance measurement, an ADC configuration for reading the analog signal generated by the ultrasonic sensor, UART communication for transmitting the measured distance information, continuous monitoring using an infinite loop, an adjustable delay of 1000 milliseconds, and error handling mechanisms. Future work could include flow control integration, IoT integration, energy optimization, user interface development, and advanced distance calibration. The proposed work lays the groundwork for a liquid-level detection system with potential applications in various domains, and future work can enhance functionality, usability, and integration with other systems.

**CHAPTER V: CONCLUSION AND FUTURE WORK**

The project focused on liquid level detection and water flow control has successfully implemented a robust system that provides accurate and real-time data on liquid levels. The system is designed to withstand various environmental conditions and deliver precise measurements, contributing to enhanced decision-making processes. The integration of reliable ultrasonic sensors, flow sensors, microcontrollers, and data processing units enhance the functionality of the system, providing a holistic view of fluid systems. Future work includes remote monitoring and control through IoT technologies, machine learning integration for predictive analysis and anomaly detection, integration with smart systems, an enhanced user interface, and energy-efficient sensors. These advancements will enable users to access real-time data and manage liquid levels remotely, forecast potential issues, and trigger proactive maintenance actions. The project lays a solid foundation for liquid level detection and water flow monitoring, and future endeavors can lead to more sophisticated, efficient, and user-friendly systems with broader applications.

**REFERENCES**

1. S. Gunde, A. K. Chikaraddi and V. P. Baligar, "IoT based flow control system using Raspberry PI," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 2017, pp. 1386-1390, doi: 10.1109/ICECDS.2017.8389671.

2. Jan, F.; Min-Allah, N.; Saeed, S.; Iqbal, S.Z.; Ahmed, R. IoT-Based Solutions to Monitor Water Level, Leakage, and Motor Control for Smart Water Tanks. Water 2022, 14, 309. https://doi.org/10.3390/w14030309

3. K. S. Shingote and P. Shahane, "Microcontroller Based Flow Control System for Canal Gates in Irrigation Canal Automation," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 796-800, doi: 10.1109/IACC.2016.152.