KOCAELI UNIVERSITY

FACULTY OF ENGINEERING

IOT BASED MANHOLE DETECTION AND MONITORING SYSTEM

C# PROGRAMMING PROJECT REPORT

Hayati Kılıç

Department: Electronics and Communication Engineering

KOCAELİ, 2023

İNGREDİENTS

İNDEX OF SHAPES

ABBREVİATİONS

IoT	:Internet of Things
API	:Application Programming Interface
NET	:Network
UART	:Universal Asynchronous Receiver Transmitter
USART	:Universal Synchronous Asynchronous Receiver Transmitter
ADC	:Analog Digital Converter
PWM	:Pulse Width Modulation
A	:Amper
V	:Volt
TIM	:Timer
IT	:Interrupt

1. LOGIN

Urban infrastructure poses myriad challenges, and among the most overlooked yet perilous are unattended manholes. The absence of proper monitoring and maintenance of these manhole covers has led to numerous accidents, endangering pedestrians, motorists, and residents alike. Addressing this critical safety concern, an innovative solution emerges in the form of the IoT Based Manhole Detection and Monitoring System.

This groundbreaking system represents a fusion of technological prowess and social responsibility. Leveraging the Internet of Things (IoT) capabilities and advanced sensor technologies, it endeavors to revolutionize the oversight of manhole covers within urban environments. The primary objective is to ensure proactive monitoring and detection of any anomalies or open manholes in real-time.

At its core, this system operates on a sophisticated network of sensors strategically positioned across cityscapes. These sensors continuously gather data pertaining to the status of manhole covers, promptly relaying any irregularities to a centralized monitoring hub. Through a seamless integration of STM32F4 microcontroller technology and C# programming, the system enables rapid data processing, analysis, and immediate alert notifications.

The significance of this system extends far beyond mere detection; it embodies a paradigm shift towards preemptive safety measures. By providing municipal authorities and stakeholders with comprehensive insights through an intuitive interface, it enables swift responses to potential hazards. This centralized platform equips decision-makers with the tools necessary for informed interventions, thereby mitigating risks and enhancing urban safety.

Furthermore, the scalability and adaptability of this IoT-based solution hold promise for implementation in diverse urban landscapes globally. Its potential impact on averting accidents stemming from unsecured manholes positions it as a cornerstone of urban safety initiatives.

In essence, the IoT Based Manhole Detection and Monitoring System stands as a testament to technological innovation aligned with societal welfare. As cities strive for resilience and safety, this system emerges as a beacon of hope, reshaping the landscape of urban safety protocols.

1.1 Literature Review

There were studies on the IoT manhole cover monitoring system, but most of these studies dealt with situations such as cover open/close, manhole temperature status, gas status. In addition to these, are there any signs of living beings inside? What's the water level? Is manhole lighting sufficient? We have taken the first step towards developing this project by making additions that answer questions such as these.

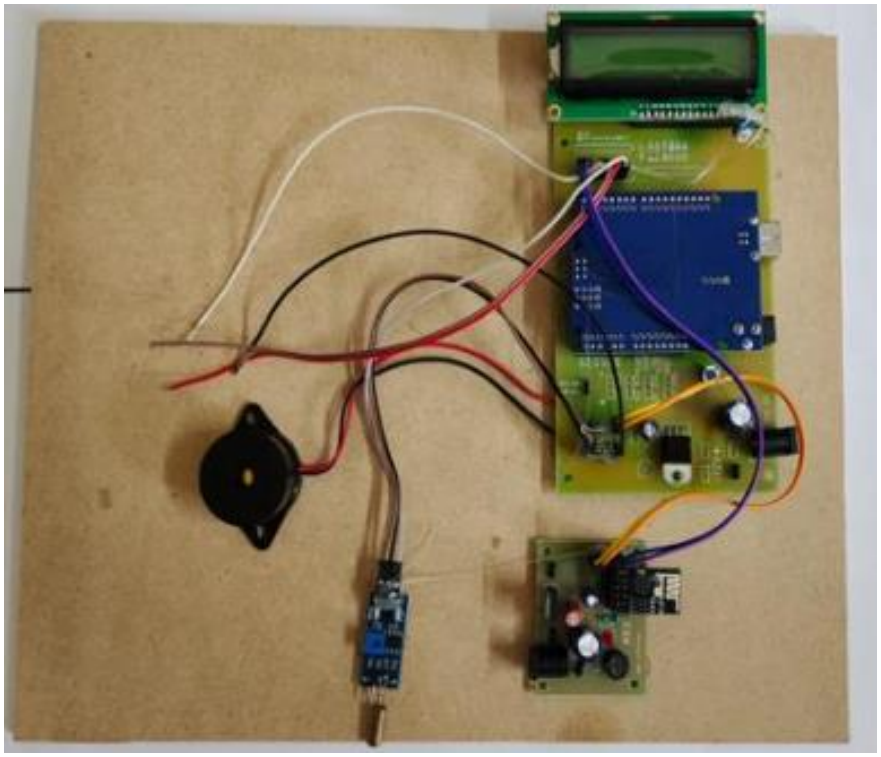

Figure 1.1 : Sample Project for Manhole Detection System

1.2 Platform Used

1.2.1 Software Platform Used

1.2.1.1 STM32CubeIDE

It is used to program the STM32F446RE Nucleo microcontroller.

1.2.1.2 Microsoft Visual Studio

It is used for the operations to be performed according to the data coming from the integrated, sending the data from the computer to the integrated and creating the user interface.

1.2.1.3 Termite

Auxiliary platform used to display data from sensors.

1.2.2 Hardware Platform Used

1.2.2.1 STM32F446-RE NUCLEO

The STM32F446-RE NUCLEO is a development board manufactured by STMicroelectronics. Belonging to the STM32 family, this board is designed based on the STM32F446RE microcontroller.

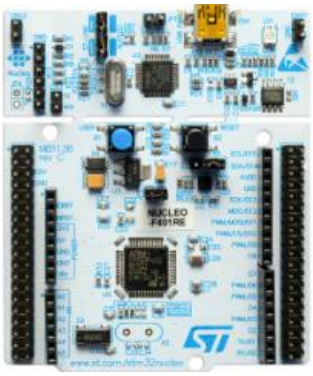

Figure 1.2.2.1 : STM32F446-RE Nucleo Microcontroller

1.2.2.2 LDR Sensor

The dark/light status of the environment is controlled with the LDR Sensor.

Figure 1.2.2.2 : LDR Sensor

1.2.2.3 Lightbulb 12V

The Bulb will be operated according to the data coming from the LDR Sensor. The bulb control will be controlled via the interface screen designed with C#.

Figure 1.2.2.3 : Lightbulb 12V

1.2.2.4 MQ2 Gas Sensor Module

The function of detecting flammable gases in the manhole will be realized with the MQ2 Gas sensor module.

Figure 1.2.2.4 : MQ2 Gas Sensor Module

1.2.2.5 Arduino Fan Module

When gas is detected, the gas balance of the manhole cover will be ensured by activating the Arduino fan module. Fan speed will be controlled from the interface screen prepared with C#.

Figure 1.2.2.5 : Arduino Fan Module

1.2.2.6 HCSR04 Ultrasonic Sensor

Water level was monitored using the HCSR04 Ultrasonic Sensor. In addition, it is aimed to determine whether there is a living creature inside with a second HCSR04 depending on this water level.

Figure 1.2.2.6 HCSR04 Ultrasonic Sensor

1.2.2.7 SG-90 Servo Motor

Depending on the water level condition, the water level will be controlled by starting the servo motor.

Figure 1.2.2.7 : SG-90 Servo Motor

1.2.2.8 Buzzer

If there is a living creature inside, the buzzer will be activated to warn people around.

Figure 1.2.2.8 : Buzzer

1.2.2.9 Adapter 12V

A 12 V adapter was used to power the bulb.

Figure 1.2.2.9 : Adapter 12V

1.2.2.10 L293D Motor Driver Module

A connection is made between the card and the 12v bulb by using the L293D motor driver module. Thanks to PWM, the brightness of the bulb is adjusted.

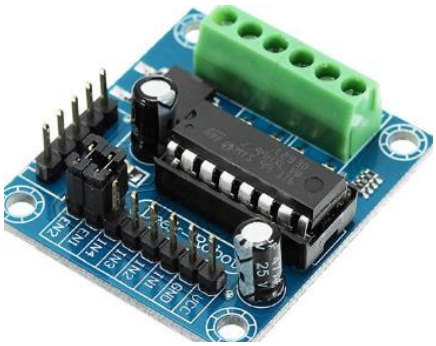

Figure 1.2.2.10 : L293D Motor Driver Module

2. REALIZATION OF THE PROJECT

2.1 Hardware Part

The implementation of the project is shown below.

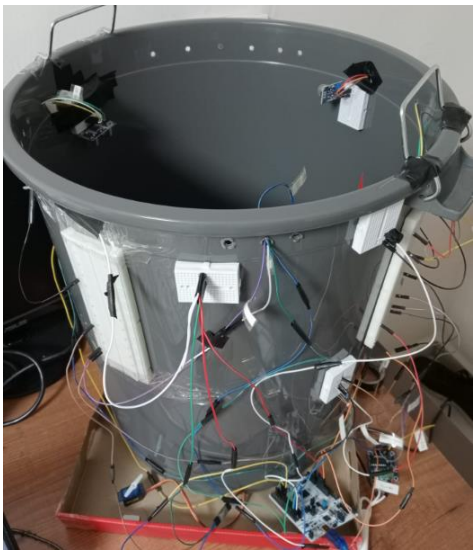

Figure 2.1 : Manhole

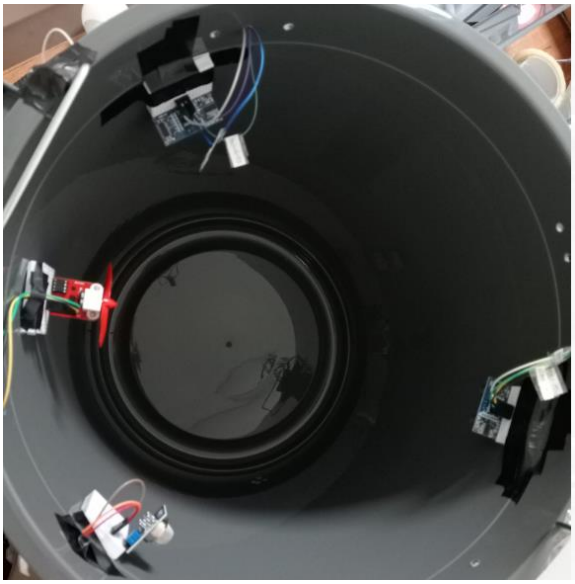

Figure 2.1.1 : Manhole II

2.2 Software Part

Important lines of code written on the embedded software side are listed below.

The code below reads analog data from the LDR sensor and converts it to digital. Returns the light_state variable based on the amount of light to send information to the C# screen.

```
uint16_t Read_LDR(void)
{
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 20);
light_value = HAL_ADC_GetValue(&hadc1);

if(light_value < 1000)
{ return 1;}
else
{return 0;}
}
```

The code below reads analog data from the MQ2 Gas sensor and converts it to digital. It returns the gas_state variable according to the amount of gas to send information to the C# screen.

```
uint16_t Read_MQ2(void)
{
HAL_ADC_Start(&hadc2);
HAL_ADC_PollForConversion(&hadc2, 20);
gas_value = HAL_ADC_GetValue(&hadc2);

if(gas_value > 450)
{return 1;}
else
{return 0;}
}
```

The following lines of code measure the water level with the HCSR04 sensor and return the water_state variable according to a certain threshold value.

```
uint32_t Read_HCSR04(void)
{
   uint32_t duration = 0;

   // TRIG pini için pulse gönderme
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
   delay_us(2);
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
   delay_us(10);
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
   // ECHO pininin yüksek olmasını bekleyerek süreyi ölçme
   uint32_t startTime = 0;
   uint32_t endTime = 0;
```

```
    // TIM4 zamanlayıcıyı başlat
    HAL_TIM_Base_Start(&htim4); // Burada htim4 sizin TIM4 yapılandırmanızı ifade etmelidir.

    // ECHO pininin yüksek olmasını bekleyerek süreyi ölçme
    while (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_5) == GPIO_PIN_RESET);
    startTime = __HAL_TIM_GET_COUNTER(&htim4); // Başlangıç zamanını al

    while (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_5) == GPIO_PIN_SET);
    endTime = __HAL_TIM_GET_COUNTER(&htim4); // Bitiş zamanını al
    // TIM4 zamanlayıcıyı durdur
    // Süreyi mikrosaniyeye dönüştürme ve mesafeyi hesaplama
    duration = endTime - startTime;
    water_value = 35 - ((duration * 0.0343) / 2);

if(water_value >= 20)
{
return 1;
}
else
{
return 0;
}

}
```

The function below decides whether there is a living creature in the manhole by comparing it with the data coming from the HCSR04 sensor that measures the water level.

```
int16_t Read_Live(uint16_t water_value)
{
  uint32_t duration = 0;

  // TRIG pini için pulse gönderme
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);
  delay_us(2);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);
  delay_us(10);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);

  // ECHO pininin yüksek olmasını bekleyerek süreyi ölçme
  uint32_t startTime = 0;
  uint32_t endTime = 0;

  // TIM4 zamanlayıcıyı başlat
  HAL_TIM_Base_Start(&htim4); // Burada htim4 sizin TIM4 yapılandırmanızı ifade etmelidir.

  // ECHO pininin yüksek olmasını bekleyerek süreyi ölçme
  while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_5) == GPIO_PIN_RESET);
  startTime = __HAL_TIM_GET_COUNTER(&htim4); // Başlangıç zamanını al

  while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_5) == GPIO_PIN_SET);
  endTime = __HAL_TIM_GET_COUNTER(&htim4); // Bitiş zamanını al

  // TIM4 zamanlayıcıyı durdur
```

```
    // Süreyi mikrosaniyeye dönüştürme ve mesafeyi hesaplama
    duration = endTime - startTime;
    water_value2 = 35 - ((duration * 0.0343) / 2);

if(water_value2 - water_value > 5)
{
return 1;
}
else
{
return 0;
}
}
```

Finally, below are the TIM IT and UART IT codes.

```
void TIM3_IRQHandler(void)
{
light_state = Read_LDR();
gas_state = Read_MQ2();
water_state = Read_HCSR04();
live_state = Read_Live(water_value);
sprintf(str,"%hu, %hu, %hu, %hu\r\n",water_state, gas_state, light_state, live_state);
HAL_UART_Transmit(&huart2, (uint8_t *)str, strlen(str), HAL_MAX_DELAY);
            HAL_TIM_IRQHandler(&htim3);
  }
```

```
void USART2_IRQHandler(void)
{
   if (huart2.Instance == USART2) {
      if(rxIndex < 50) {
         rxBuffer[rxIndex++] = huart2.Instance->DR & 0xFF;
         if(rxBuffer[rxIndex - 1] == '\n') {
            rxBuffer[rxIndex - 1] = '\0';
            // Gelen verileri işleyin
            char *timePtr = strstr((char *)rxBuffer, "Buzzer Time =");
            if(timePtr != NULL) {
               sscanf(timePtr, "Buzzer Time = %hu", &Buzzer_Time_Value);
            }
            char *servoSpeedPtr = strstr((char *)rxBuffer, "Servo Speed =");
            if(servoSpeedPtr != NULL) {
               sscanf(servoSpeedPtr, "Servo Speed = %hu", &Servo_Speed_Value);
            }
            char *fanSpeedPtr = strstr((char *)rxBuffer, "Fan Speed =");
            if(fanSpeedPtr != NULL) {
               sscanf(fanSpeedPtr, "Fan Speed = %hu", &Fan_Speed_Value);
            }
            char *lightIntensityPtr = strstr((char *)rxBuffer, "Light Intensity = ");
            if(lightIntensityPtr != NULL) {
             sscanf(lightIntensityPtr, "Light Intensity = %hu", &Light_Intensity_Value);
            }
            rxIndex = 0;
            memset(rxBuffer, 0, sizeof(rxBuffer));
         }
      }
   }
   HAL_UART_Receive_IT(&huart2, (uint8_t *)rxBuffer + rxIndex, 1);
// Tekrar veri alma işlemi başlat
  HAL_UART_IRQHandler(&huart2); }
```

Figure 2.2 : Interface Application Designed in C#

Below are some of the C# codes.

```
1 başvuru
private void btnOpen_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.PortName = cBoxCOMPORT.Text;
        serialPort1.BaudRate = Convert.ToInt32(cBoxBAUDRATE.Text);
        serialPort1.DataBits = Convert.ToInt32(cBoxDATABITS.Text);
        serialPort1.StopBits = (StopBits)Enum.Parse(typeof(StopBits), cBoxSTOPBITS.Text);
        serialPort1.Parity = (Parity)Enum.Parse(typeof(Parity), cBoxPARITYBITS.Text);

        serialPort1.Open();
        progressBar1.Value = 100;
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message,"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Figure 2.2.1 : Buton Open Click Codes

```
2 başvuru
private void HandleReceivedData(string receivedData)
{
    string[] dataValues = receivedData.Split(',');
    if(dataValues.Length >= 4)
    {
        try
        {
            waterValue = int.Parse(dataValues[0]);
            gasValue = int.Parse(dataValues[1]);
            lightValue = int.Parse(dataValues[2]);
            soundValue = int.Parse(dataValues[3]);

            CheckStatus(waterValue, gasValue, lightValue, soundValue);
        }
        catch(Exception ex)
        {
            MessageBox.Show("Data processing error: " + ex.Message);
        }
    }
}
```

Figure 2.2.2 : Handle Received Data Codes

```
1 başvuru
private void timer1_Tick(object sender, EventArgs e)
{
    if(serialPort1.IsOpen && serialPort1.BytesToRead > 0)
    {
        string receivedData = serialPort1.ReadLine();
        HandleReceivedData(receivedData);
    }
}
```

Figure 2.2.3 : Timer 1 Codes

CONCLUSIONS AND RECOMMENDATIONS

This study aimed to evaluate the performance of the IoT-Based Manhole Detection and Monitoring System in enhancing manhole safety. The research yielded the following key findings:

1. Effective Monitoring and Swift Intervention: The IoT technology employed provided an effective monitoring system with swift intervention capabilities for manhole safety. The system demonstrated successful detection of open manholes and preemptive hazard prevention.

2. User-Friendly Interface and Feedback Mechanism: The system's user interface facilitated instantaneous data delivery and alert mechanisms to relevant stakeholders, streamlining prompt and informed decision-making.

Based on this study's findings, the following recommendations are proposed:

1. Public Awareness Initiatives: Organizing educational campaigns to raise public awareness regarding manhole safety is crucial. Such efforts could significantly contribute to accident prevention.

2. Expansion of Technological Implementation: Wider deployment and observation of IoT-based systems in more urban areas are essential to enhance city safety standards.

3. Data Security Enhancements: Regular review and updates of data security and system update procedures are imperative to enhance system reliability.

These recommendations serve as fundamental steps toward progress in manhole safety and the more effective utilization of the IoT-Based Manhole Detection and Monitoring System.

REFERENCES

[1] https://nevonprojects.com/iot-based-manhole-detection-and-monitoring-system/

[2] IoT based Manhole Detection and Monitoring System. Varun Krishna Nallamothu, Saahith Medidi, Swetha Priyanka Jannu.

[3]https://www.ijraset.com/research-paper/iot-based-manhole-detection-and-monitoring-system

[4] https://www.sciencedirect.com/science/article/abs/pii/S2214785321079967

[5]Manhole cover monitoring system over IOT. Wesam Moneer Rasheed, Raed Abdulla, Low Yee San.

[6] Man Hole Detection and Monitoring System Using IOT. Mr.ManeHarshavardhan Vijay ,Mr.Nimbaler Swapnil Sanjay , ChougulePushpraj Babaso , Mr.Ghatage Abhishek Dundappa , Ms.Saundatte M .G.

[7] User Access Control System based on ESP32 Technology. Vinícius Penckowski.

[8] A Review on Manhole Monitoring System. Anusha N. , Niveditha V. , Pooja , Preethu S. , Asst Prof. Girijamba D. L.

[9] https://iotdesignpro.com/projects/iot-manhole-monitoring-system

[10] Small Single Board Computers based Smart Manhole Monitoring and Detection System. N Vikram, Ramakrishnan Raman, J. Jagan Babu, E. Srividhya.

[11] An IoT Based Proposed System for Monitoring Manhole in Context of Bangladesh. Saadnoor Salehin, Syeda Sabrina Akter, Anika Ibnat, Tasmiah Tamzid Anannya, Nurun Nahar Liya, Manisha Paramita, Md Mahboob Karim.