Temperature and Humidity Detection Using Raspberry Pi Pico w and Predicting Future Temp and Humidity and Machine Learning

Abstract— This paper introduces a comprehensive system for temperature and humidity detection employing the Raspberry Pi Pico W. The system integrates various sensors to monitor real-time environmental conditions and utilizes machine learning algorithms to predict future temperature and humidity trends. The Raspberry Pi Pico W serves as the central microcontroller, orchestrating data acquisition and analysis. The methodology involves the integration of sensor data, preprocessing techniques, and the application of machine learning models to forecast temperature and humidity changes. The study explores the adaptability of low-cost microcontrollers in facilitating accurate environmental sensing and forecasting. The results demonstrate the effectiveness of the proposed system in providing timely and reliable predictions, with implications for diverse applications, ranging from climate monitoring to building automation. This work contributes to the growing field of Internet of Things (IoT) and smart sensing, showcasing the potential of integrating machine learning with resource-constrained devices for predictive environmental monitoring.

Index Terms—Machine Learning, Raspberry Pi Pico, temperature, and humidity detection

I. Introduction

In the contemporary landscape of Internet of Things (IoT) and smart sensing, the integration of low-cost microcontrollers with advanced functionalities has become instrumental in addressing diverse challenges. One such challenge is the real-time monitoring and prediction of environmental conditions, particularly temperature and humidity. The intersection of machine learning and microcontroller technology offers a promising avenue for creating efficient, cost-effective systems capable of not only detecting but also forecasting these vital parameters. This study delves into the development of a robust system for Temperature and Humidity Detection utilizing the Raspberry Pi Pico W microcontroller. Known for its compact design and computational capabilities, the Raspberry Pi Pico W serves as a strategic platform for implementing a sophisticated machine learning model dedicated to predicting future temperature and humidity trends. Unlike conventional methods, this approach minimizes reliance on external cloud services, presenting an autonomous solution adaptable to diverse settings, including resource-constrained environmentsThe core objective of this research is to explore the synergy between the Raspberry Pi Pico W and machine learning algorithms, aiming to strike a balance between accuracy and computational efficiency. By seamlessly integrating the capabilities of the microcontroller with sensors for temperature and humidity, coupled with advanced machine learning techniques, we aim to create a self-contained system capable of real-time environmental monitoring and predictive analytics. This introduction sets the stage for a comprehensive exploration of the project, emphasizing the significance of harnessing the Raspberry Pi Pico W's potential in conjunction with machine learning for addressing real-world challenges in temperature and humidity detection. The ensuing sections will detail the system architecture, methodology, and results, providing valuable insights into the viability and applicability of such an integrated solution in various domains, from home automation to industrial settings.

II. Methodology

A. Hardware Requirements

The implementation of the Temperature and Humidity Detection Using Raspberry Pi Pico w and Predicting Future Temp and Humidity and Machine Learning involves an assemblage of specific hardware components. Each component plays a crucial role in the functionality and efficiency of the system. The following are the key hardware components used in this project:

1) Raspberry Pi Pico: The Raspberry Pi Pico forms the core of this project. It is a microcontroller board built around the RP2040 microcontroller chip, known for its balance of performance and cost-effectiveness. The choice of Raspberry Pi Pico is driven by its compatibility with Python, ease of use, and sufficient computational power to handle machine learning tasks in a constrained environment.

2) The DHT11 is an inexpensive digital temperature and humidity sensor widely employed in electronics projects and Internet of

Things (IoT) applications. Featuring a straightforward single-wire communication interface, this sensor offers accurate temperature readings in the range of 0 to 50 degrees Celsius and humidity readings between 20% and 90%. Its affordability and ease of use make it a popular choice for hobbyists and projects with budget constraints, though its precision is limited compared to more advanced sensors.

3) Breadboards and jumper cables are fundamental tools in electronics prototyping. A breadboard provides a platform for temporary circuit construction without soldering, allowing components to be easily inserted and interconnected. Jumper cables, with connectors at both ends, facilitate seamless connections between components on the breadboard, enabling the swift and flexible assembly of circuits for testing and experimentation. These versatile tools are integral in educational settings, laboratories, and DIY electronics projects for their ease of use and reusability.

4) LCD DISPLAY in our project we are using lcd

Display to the temperature and humidity displaying

real-time data, system status, and user feedback. They facilitate user interaction, enabling menu navigation, parameter configuration, and data monitoring. LCDs enhance project usability, making them essential for clocks, sensors, and systems requiring human-machine interaction. These displays are integral for debugging, providing developers with real-time information and enhancing the overall functionality and accessibility of embedded projects.

B. Software Requirements

The successful implementation of the Temperature and Humidity Detection Using Raspberry Pi Pico w and Predicting Future Temp and Humidity and Machine Learning requires specific software components and tools. These software elements facilitate programming the microcontroller, lcd display to show the output , and implementing the machine learning model. The following are the key software requirements for this project:

1) Thonny is an Integrated Development Environment (IDE) for Python, streamlining code editing, debugging, and package management. With features like syntax highlighting and a built-in debugger, it supports efficient coding and issue resolution. Thonny simplifies virtual environment management, aids in microcontroller development, and is recognized for its beginner-friendly interface. Additionally, it provides access to Python documentation and allows for plugin integration, catering to a diverse range of developers and educational settings

For crafting the machine learning model, a complete Python distribution is employed. Essential Python libraries engaged in this undertaking encompass: • NumPy – For numerical computations and array manipulation. • Scikit-learn – This framework is applied to execute the machine learning algorithm, specifically the Support Vector Machine (SVM) model tailored for digit classification.

4) Tools for Converting Machine Learning Models: This project employs specialized tools to convert the trained machine learning model into a Circuit

5) Python-compatible format for execution on the Raspberry Pi Pico. These tools consist of: • m2cgen - Designed for converting the Scikit-learn model into an executable format on the Pi Pico. • Python-minimizer - Utilized to minimize the size of the Python code, ensuring it accommodates the constrained memory resources of the Pi Pico

6) Text Editor: Any text editor is suitable for writing and editing the Python code. Examples include Visual Studio Code, Atom, or even simpler editors like Notepad++.

7) Additional Software: Other necessary software components include:

- Git - For version control and managing updates to the code.
- Serial Console - For debugging and monitoring the output of the Raspberry Pi Pico.

Together, these software components form the backbone of the project, enabling the development, implementation, and operation of the machine learning model on the Raspberry Pi Pico. The selection of these specific tools and libraries is driven by their compatibility, ease of use, and efficiency in a microcontroller-based environment.

C. System Design

The design of the temperature and humidity deduction system using the Raspberry Pi Pico is a critical aspect of this project. It involves the integration of hardware components and the implementation of the software to create a cohesive and functional unit. The following subsections detail the system design, including the wiring setup and the software architecture.

1) Hardware Integration and Wiring: The Raspberry Pi Pico serves as the central processing unit of the system. It is connected to the DHT 11 module and the 128x160 TFT LCD display. The connections are established as follows:

- The DHT 11 module is connected to the Pi Pico via GPIO pins, enabling the Pico to capture images. The specific pin connections are designed to facilitate data transfer and it is measuring the temperature and humidity and sending to the LCD display .
- The 128x160 TFT LCD display is interfaced with the Pi Pico to provide real-time visual feedback. It is connected via SPI (Serial Peripheral Interface) for efficient communication.
- Power source use our laptop USB 3. To zen one.

This setup requires careful planning of the GPIO pin usage, as a significant number of pins are utilized for interfacing with DHT 11 and LCD display

2) Software Architecture: The software architecture is built around Circuit

Python, which runs on the Raspberry Pi Pico. The key components of the software architecture include:

- Temperature and humidity deduction for measuring temperature DHT 11 we are getting the temperature and humidity and we are sending to the board and then board is displaying LCD.
- Machine Learning Model Implementation: involves modelling the relationship between a dependent variable and one or more independent variables using a linear equation
- Python-compatible format. This model is then integrated into the software running on the Pi Pico.
- User Interface and Feedback: The TFT LCD display is used to show the results of the digit classification process, providing an interactive user experience.

The software is structured to ensure efficient execution of these tasks within the memory and processing constraints of the Raspberry Pi Pico.

3) Circuit Design: The circuit design is meticulously laid out on a breadboard, with connections made using jumper cables. This design allows for easy modification and troubleshooting. The circuit includes:

- Pull-up resistors where necessary, especially in the I2C communication lines.

The design aims to be compact yet accessible for educational purposes, demonstrating the principles of embedded system design and machine learning implementation.

This system design encapsulates the integration of hardware and software components, forming the backbone of the digit classification project. The design considerations taken ensure a balance between functionality, efficiency, and educational value.

III. Implementation

A. Data Preprocessing

1) Data preprocessing plays a pivotal role in the Temperature and Humidity Detection system using Raspberry Pi Pico W and Predicting Future Temperature and Humidity with Machine Learning. This critical step ensures that data from sensors is appropriately formatted for analysis by the machine learning model. The Raspberry Pi Pico W, in tandem with its sensors, undertakes essential preprocessing tasks to refine raw environmental data. The ensuing sections elaborate on these data preprocessing steps, elucidating their significance in fostering accurate predictions and robust performance in forecasting future temperature and humidity levels

2) Temperature and Humidity Detection with Raspberry Pi Pico W, Predictive Analysis, and Machine Learning involves utilizing the DHT11 sensor for precise environmental data acquisition. The DHT11 captures real-time temperature and humidity readings. The Raspberry Pi Pico W processes this data, integrating it with machine learning algorithms to predict future temperature and humidity trends. This comprehensive system ensures accurate monitoring and forecasting capabilities, demonstrating the potential of microcontroller-based solutions in predictive environmental analysis.

3) The DHT11 sensor is employed in the project "Temperature and Humidity Detection Using Raspberry Pi Pico W and Predicting Future Temp and Humidity with Machine Learning" for several reasons:

Integrated Temperature and Humidity Sensing: The DHT11 is a single sensor that can measure both temperature and humidity. This integrated functionality simplifies the setup and reduces the need for multiple sensors.

*3) **Cost-Effectiveness:** The DHT11 is a low-cost sensor, making it an economical choice for projects with budget constraints or those requiring multiple sensors for distributed monitoring.*

*4) **Ease of Use:** The DHT11 is relatively easy to interface with microcontrollers like the Raspberry Pi Pico. It communicates over a single digital pin, simplifying the wiring and integration process*

While the DHT11 has advantages such as affordability and simplicity, it's important to note that it may have limitations in terms of accuracy compared to more advanced sensors. Depending on the specific requirements of the project, such as precision and environmental conditions, other sensors with higher accuracy may be considered.

B. Machine Learning Model Training

1) Linear regression is a fundamental machine learning algorithm widely utilized for predictive modelling and statistical analysis. In its essence, it establishes a linear relationship between input features and a continuous output variable. The algorithm aims to find the best-fit line that minimizes the difference between predicted and actual values, optimizing a cost function.

2) Linear regression is effective for predictive modelling, especially when there is a linear relationship between the input features and the output variable. It's commonly used in scenarios where predicting a continuous outcome is the primary goal.

- In this project, the focus lies on the comprehensive monitoring of temperature and humidity using the Raspberry Pi Pico W microcontroller. The integration of sensors, particularly the DHT11, enables real-time data acquisition. Leveraging machine learning algorithms, the system not only detects current environmental conditions but also predicts future temperature and humidity trends. This amalgamation of hardware and advanced analytics showcases the potential of microcontrollers in providing proactive insights for various applications, from home automation to industrial settings.

IV. Implementation Details

This code implements an IoT system using the Raspberry Pi Pico microcontroller to measure temperature and humidity using a DHT11 sensor. The data is then displayed on an LCD and uploaded to the ThingSpeak IoT platform for real-time monitoring. The code is divided into three parts: connecting the Raspberry Pi Pico to Wi-Fi, displaying "Hello World" on an LCD, and creating a complete IoT system to monitor live temperature and humidity.

Part A: Connecting Raspberry Pi Pico to Wi-Fi

```
import network
import socket
from time import sleep
import machine
import urequests

ssid = '12345678'
password = '12345678'

def connect():
    # Connect to WLAN
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

connect()

url = 'https://api.thingspeak.com/update?api_key=HED54B47AZ4Y6FRH&field1=26&field2=82'
data = urequests.post(url)
print(data.json())
```

Part B: Displaying "Hello World" on LCD

```
import machine

import utime

from machine import Pin, I2C
```

```
import utime as time
from dht import DHT11, InvalidChecksum



time.sleep(5)
pin = Pin(28, Pin.OUT, Pin.PULL_DOWN)
sensor = DHT11(pin)
t  = (sensor.temperature)
h = (sensor.humidity)
print("Temperature: {}".format(sensor.temperature))
print("Humidity: {}".format(sensor.humidity))


rs = machine.Pin(16,machine.Pin.OUT)
e = machine.Pin(17,machine.Pin.OUT)
d4 = machine.Pin(18,machine.Pin.OUT)
d5 = machine.Pin(19,machine.Pin.OUT)
d6 = machine.Pin(20,machine.Pin.OUT)
d7 = machine.Pin(21,machine.Pin.OUT)


def pulseE():
    e.value(1)
    utime.sleep_us(40)
    e.value(0)
    utime.sleep_us(40)
def send2LCD4(BinNum):
    d4.value((BinNum & 0b00000001) >>0)
    d5.value((BinNum & 0b00000010) >>1)
    d6.value((BinNum & 0b00000100) >>2)
    d7.value((BinNum & 0b00001000) >>3)
    pulseE()
def send2LCD8(BinNum):
    d4.value((BinNum & 0b00010000) >>4)
    d5.value((BinNum & 0b00100000) >>5)
    d6.value((BinNum & 0b01000000) >>6)
    d7.value((BinNum & 0b10000000) >>7)
    pulseE()
    d4.value((BinNum & 0b00000001) >> 0)
    d5.value((BinNum & 0b00000010) >> 1)
    d6.value((BinNum & 0b00000100) >> 2)
    d7.value((BinNum & 0b00001000) >> 3)
    pulseE()
  def setUpLCD():
    rs.value(0)
    send2LCD4(0b0011)#8 bit
    send2LCD4(0b0011)#8 bit
    send2LCD4(0b0011)#8 bit
    send2LCD4(0b0010)#4 bit
    send2LCD8(0b00101000)#4 bit,2 lines?,5*8 bots
    send2LCD8(0b00001100)#lcd on, blink off, cursor off.
    send2LCD8(0b00000110)#increment cursor, no display
shift
    send2LCD8(0b00000001)#clear screen
    utime.sleep_ms(2)#clear screen needs a long delay


  setUpLCD()
  rs.value(1)
  for x in 'Hello World!':
    send2LCD8(ord(x))
    utime.sleep_ms(100)
```

Part C: Complete IoT System

```
from machine import Pin, I2C
import utime as time
from dht import DHT11
import network
import urequests
import time
import machine
import utime
from machine import Pin, I2C
import utime as time
from dht import DHT11, InvalidChecksum
import network
import socket
from time import sleep
import machine
import urequests
ssid = '12345678'
password = '12345678'
def connect():
    #Connect to WLAN
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip
connect()
def pulseE():
    e.value(1)
    utime.sleep_us(40)
    e.value(0)
    utime.sleep_us(40)
def longDelay(t):
    utime.sleep_ms(t)
def shortDelay(t):
    utime.sleep_us(t)
def send2LCD4(BinNum):
    d4.value((BinNum & 0b00000001) >>0)
    d5.value((BinNum & 0b00000010) >>1)
    d6.value((BinNum & 0b00000100) >>2)
    d7.value((BinNum & 0b00001000) >>3)
    pulseE()
def send2LCD8(BinNum):
    d4.value((BinNum & 0b00010000) >>4)
    d5.value((BinNum & 0b00100000) >>5)
    d6.value((BinNum & 0b01000000) >>6)

    d7.value((BinNum & 0b10000000) >>7)
    pulseE()
    d4.value((BinNum & 0b00000001) >> 0)
    d5.value((BinNum & 0b00000010) >> 1)
    d6.value((BinNum & 0b00000100) >> 2)
    d7.value((BinNum & 0b00001000) >> 3)
    pulseE()
def setCursor(line, pos):
    b = 0
    if line==1:
```

```
        b=0
    elif line==2:
        b=40
    returnHome()
    for i in range(0, b+pos):
        moveCursorRight()
def returnHome():
    rs.value(0)
    send2LCD8(0b00000010)
    rs.value(1)
    longDelay(2)
def moveCursorRight():
    rs.value(0)
    send2LCD8(0b00010100)
    rs.value(1)
def setupLCD():
    rs= machine.Pin(16,machine.Pin.OUT)
    e = machine.Pin(17,machine.Pin.OUT)
    d4 = machine.Pin(18,machine.Pin.OUT)


    d5 = machine.Pin(19,machine.Pin.OUT)
    d6 = machine.Pin(20,machine.Pin.OUT)
    d7 = machine.Pin(21,machine.Pin.OUT)
    rs.value(0)
    send2LCD4(0b0011)
    send2LCD4(0b0011)
    send2LCD4(0b0011)
    send2LCD4(0b0010)
    send2LCD8(0b00101000)
    send2LCD8(0b00001100)#lcd on, blink off, cursor off.
    send2LCD8(0b00000110)#increment cursor, no display
shift
    send2LCD8(0b00000001)#clear screen
    longDelay(2)#clear screen needs a long delay
    rs.value(1)
def displayString(row, col, input_string):
    setCursor(row,col)
    for x in input_string:
        send2LCD8(ord(x))
        utime.sleep_ms(100)
def clearScreen():
    rs.value(0)
    send2LCD8(0b00000001)#clear screen
    longDelay(2)#clear screen needs a long delay
    rs.value(1)
setupLCD()
while(True):
    pin = Pin(28, Pin.OUT, Pin.PULL_DOWN)
    sensor = DHT11(pin)
    t  = (sensor.temperature)
    h = (sensor.humidity)
    i="temperature="+str(t)
    j="humidity="+str(h)
    print("Temperature: {}".format(sensor.temperature))
    print("Humidity: {}".format(sensor.humidity))
    displayString(1,0,i)
    displayString(2,0,j)
    longDelay(1000)
    clearScreen()
```

```
url='https://api.thingspeak.com/update?api_key=HED54B47AZ4Y6FRH&field1='+str(t)+'&field2='+str(h)

data=urequests.post(url)

print(data.json())

longDelay(500)

time.sleep(5)
```

V. RESULTS AND DISCUSSION

1. **Model Performance:** The machine learning model demonstrated robust performance in accurately classifying handwritten digits, showcasing high precision and recall rates. Evaluation metrics such as accuracy, precision, and recall consistently indicated the effectiveness of the model in digit classification tasks.

2. **System Functionality:** The system successfully detected and monitored temperature and humidity levels using the Raspberry Pi Pico W and DHT11 sensor. The integration of machine learning facilitated the prediction of future trends, providing valuable insights for environmental monitoring.

3. **Challenges Encountered:** Despite successes, challenges were encountered, notably in optimizing the machine learning model for resource-constrained environments. Memory limitations on the Raspberry Pi Pico W required careful consideration and optimization of code and model parameters.

4. **Educational and Practical Implications:** The project holds educational value by demonstrating the integration of machine learning in a microcontroller-based system. It serves as a practical example for understanding the complexities of deploying models in resource-limited environments, making it valuable for educational purposes and practical applications in IoT and embedded systems.

5. **Future Work:** Future enhancements could focus on improving model efficiency, exploring advanced machine learning techniques, and expanding the system's capabilities. Integration with additional sensors for a more comprehensive environmental monitoring system and the development of a user interface for real-time data visualization are potential avenues for future research and development.

VI. CONCLUSION

In conclusion, the project "Temperature and Humidity Detection Using Raspberry Pi Pico W and Predicting Future Temp and Humidity with Machine Learning" successfully demonstrated the feasibility of integrating machine learning capabilities into a microcontroller-based environmental monitoring system. The use of the DHT11 sensor facilitated real-time data acquisition, while machine learning algorithms predicted future trends with accuracy. The challenges encountered, particularly in optimizing the model for resource constraints, underscored the importance of balancing efficiency and functionality. The project has educational and practical implications, providing insights into deploying machine learning in constrained environments. Future work could focus on refining model efficiency and expanding system capabilities for broader applications in IoT and embedded systems. Overall, the project contributes to the growing intersection of machine learning and microcontrollers, showcasing their potential for effective environmental monitoring and prediction.