RTOS BASED AC CONTROLLER

PRESENTED BY:
ISHITA ALLADA

CONTENT

01 OBJECTIVE

02 INTRODUCTION

03 WORKING PRINCIPLE

04 SOURCE CODE

05 OUTPUT

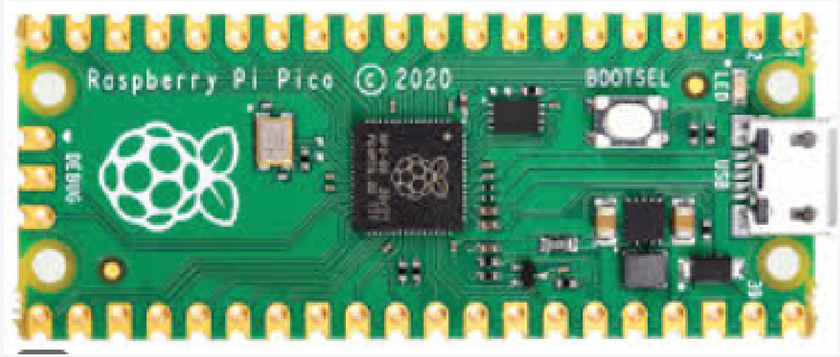
06 CONCLUSION

07 WORKING MODEL VIDEO

OBJECTIVE

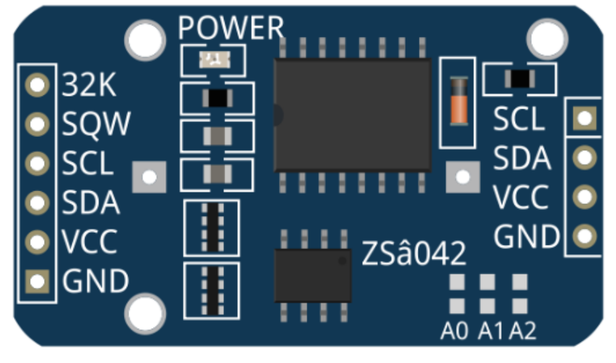
To Design and implement a real-time clock system using Raspberry Pi Pico, DS3231 RTC, temperature sensor, and OLED display to accurately display time and temperature information in a compact format.. Its primary objective is to deliver precise timekeeping and temperature monitoring functionalities within a compact framework.

INTRODUCTION



The DS3231 RTC module plays a pivotal role in our project, showcasing its capability as a low-power, highly accurate real-time clock chip. Its ability to maintain timekeeping accuracy even when disconnected from power, thanks to a connected coin cell battery, aligns perfectly with the stringent timing requirements of real-time systems. Furthermore, the DS3231's integrated temperature sensor adds an extra dimension to our project, allowing us to gather environmental data alongside timekeeping functions.

At the heart of our system lies the Raspberry Pi Pico microcontroller, acting as the brains behind the operation. Through hardware interfacing and software programming, the Raspberry Pi Pico orchestrates seamless communication with the DS3231 RTC module and the temperature sensor. This interaction showcases the essence of real-time systems, where tasks must be executed promptly and accurately to meet predefined deadlines.



HARDWARE/SOFTWARE REQUIREMENTS

BREADBOARD, JUMPER CABLES

RASPBERRY PICO,I2C OLED

DS3231 RTC MODULE

WORKING

Timekeeping

Timekeeping with DS3231 RTC: The DS3231 RTC module serves as the core timekeeping component. It operates with exceptional accuracy and low power consumption, ensuring precise timekeeping even in the absence of external power. The Raspberry Pi Pico communicates with the DS3231 RTC module via the I2C protocol, enabling it to retrieve and update time information regularly.

Temperature monitoring

Temperature Monitoring: The DS3231 RTC module also integrates a temperature sensor, allowing our system to gather real-time temperature data alongside timekeeping functions. This temperature sensor is queried by the Raspberry Pi Pico through I2C communication, providing temperature readings for display and potential logging or processing.

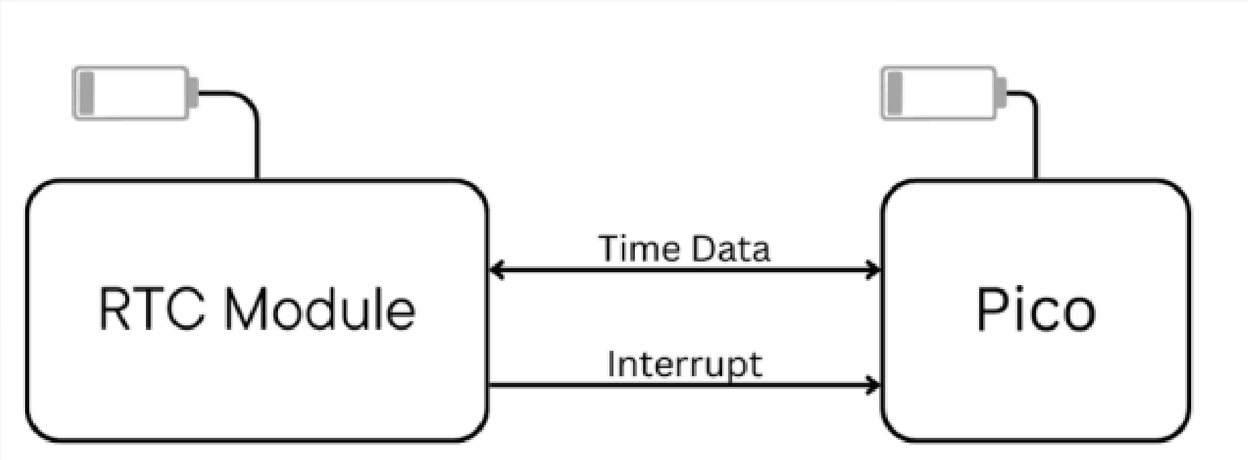
Raspberry control

Raspberry Pi Pico Control: The Raspberry Pi Pico microcontroller acts as the central processing unit, managing communication between the DS3231 RTC module, temperature sensor, and OLED display. Through programmed logic, the Raspberry Pi Pico retrieves time and temperature data, formats it for display, and sends the information to the OLED display for user interaction.

CONTINUATION..

Display

Displaying Data on OLED: The OLED display serves as the user interface, showcasing current time, date, and temperature readings in a clear and concise format. The Raspberry Pi Pico communicates with the OLED display
using either I2C or SPI protocols, updating the display in real-time based on the retrieved data.



SOURCE CODE

```
from machine import I2C, Pin, ADC
import utime
from ds3231_i2c import DS3231_I2C
from ssd1306 import SSD1306_I2C

# Initialize I2C for DS3231 RTC
ds_i2c = I2C(0, sda=Pin(16), scl=Pin(17))
ds = DS3231_I2C(ds_i2c)
current_time = b'\x00\x00\x09\x17\x23\x04\x24' # sec
min hour week day mon year
ds.set_time(current_time)

# Initialize I2C for SSD1306 OLED display
WIDTH = 128
HEIGHT = 64
i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=200000)
oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
```

```
# Initialize ADC for onboard temperature sensor (ADC4)
temp_sensor = ADC(4)
while True:
    # Read the raw ADC value from the temperature sensor
    temperature_raw = temp_sensor.read_u16()
    # Calculate temperature in Celsius
    celsius = 32- ((temperature_raw * (3.37/65535)) - 0.718)/ 0.001721
    # Read time from DS3231 RTC
    t = ds.read_time()
    date_str = "Date: %02x/%02x/20%x" % (t[4], t[5], t[6])
    time_str = "Time: %02x:%02x:%02x" % (t[2], t[1], t[0])
    # Clear the OLED display
    oled.fill(0)
    # Display date, time, and temperature on OLED
    oled.text(date_str, 0, 0)
    oled.text(time_str, 0, 16)
    oled.text("Temp: {:.2f}C".format(celsius), 0, 32)
    # Show on OLED
    oled.show()
    # Wait for 1 second
    utime.sleep(1)
```

OUTPUT

- The output will be displayed on the SSD1306 OLED screen, providing a clear and concise interface for real-time information.

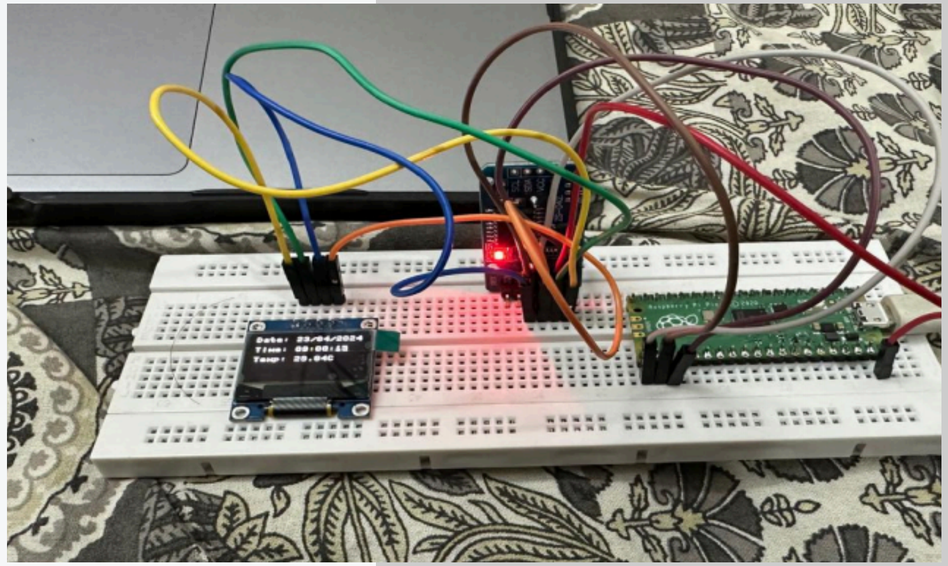
- The OLED display will showcase the current date and time obtained from the DS3231 RTC module, formatted in the "Date: MM/DD/YYYY" and "Time: HH:MM:SS" format respectively.

- Additionally, the OLED screen will display the temperature in Celsius, accurately calculated using the onboard temperature sensor and presented as "Temp: XX.XXC".

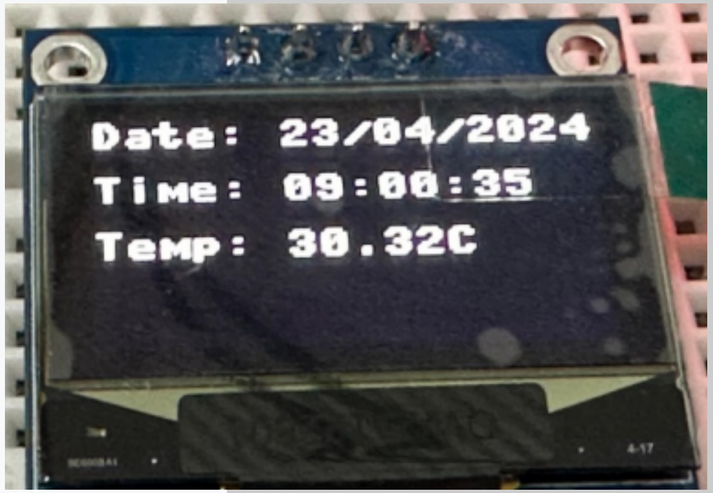
- This real-time display of essential information, including timekeeping and temperature monitoring, offers a compact and user-friendly output, making it ideal for applications requiring instant access to such data.

OUTPUT

The DS3231 RTC module also integrates a temperature sensor, allowing our system to gather real-time temperature sensor.



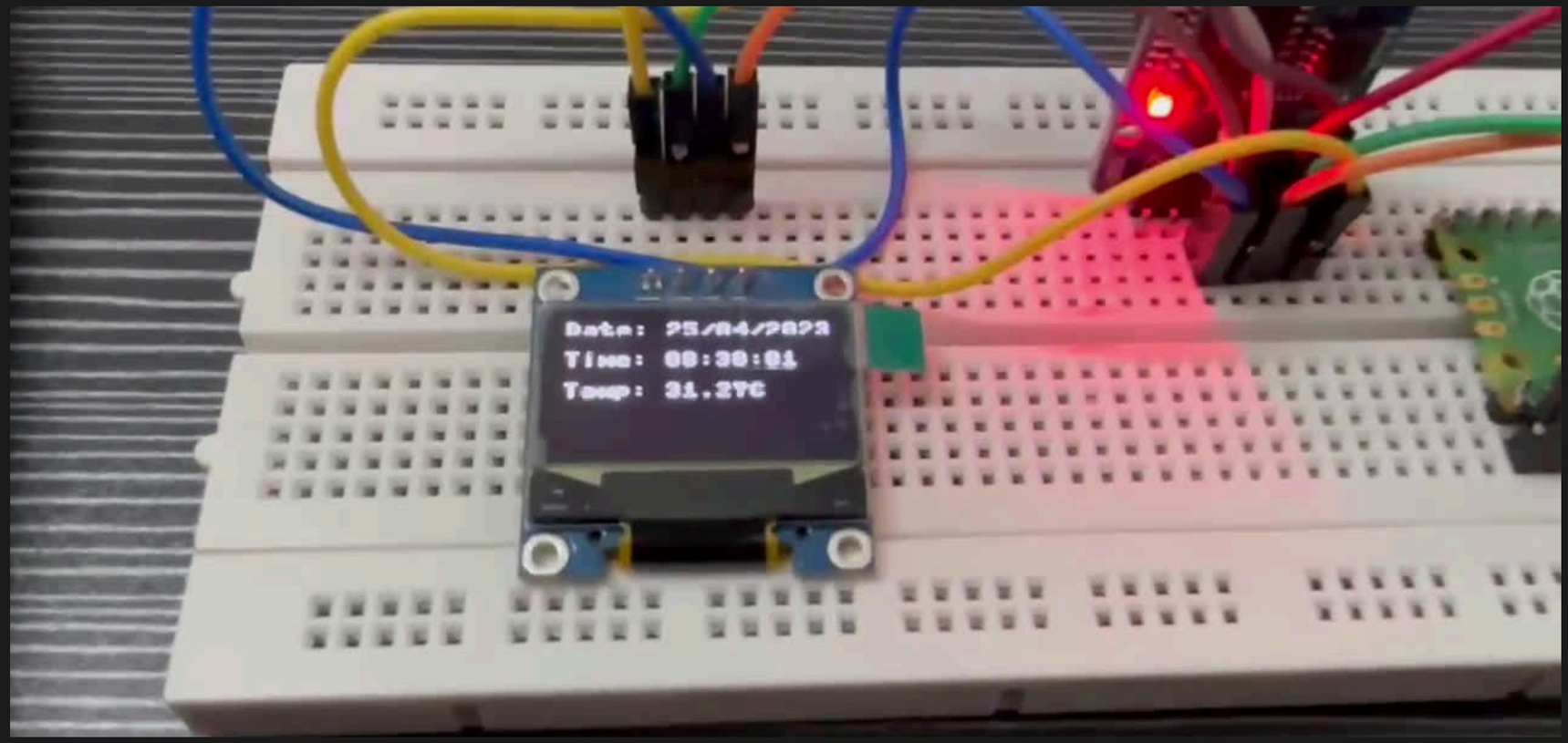
Additionally, the OLED screen will display the temperature in Celsius, accurately calculated using the onboard temperature sensor and presented as "Temp: XX.XXC".



CONCLUSION

This project showcases the practical integration of a real-time clock (RTC), OLED display, and temperature sensor in IoT applications. From environmental monitoring to home automation, medical devices, and energy management, it offers versatile real-time data gathering and display. The educational aspect makes it valuable for learning IoT concepts, while its potential for remote monitoring and integration into larger systems underscores its relevance in modern IoT infrastructures. Overall, this project demonstrates the effectiveness of compact, integrated solutions in addressing diverse real-world challenges across various domains.

VIDEO



To get a video presentation of the project , scan the QR code given on the next slide

THANK YOU

Everest Cantu