Solar-Powered Smart Sensing System: Design and Implementation with Microcontroller and Cloud Integration

Semih Isa Cirik
Graduate School of Informatics
Gazi University
Ankara, Turkey
sisa.cirik@gazi.edu.tr

***Abstract*–This paper presents the design and implementation of a solar-powered smart sensing system using a Raspberry Pi Pico microcontroller unit (MCU) with a dual-core Arm Cortex M0+ processor and Amazon Web Services (AWS) IoT platform. The system, powered by a solar panel and an 18650 lithium battery, periodically collects data from sensors (such as the DHT11 temperature and humidity sensor) and transmits it via the MQTT protocol to AWS IoT Core for analysis using services like AWS Lambda and DynamoDB. The paper discusses the components used, including a 6V 250mA solar panel, a 3.7V 2200mAh lithium battery, TP4056 charging module, and Schottky diodes, as well as the system's assembly and coding with MicroPython. The system enabled continuous 24-hour operation with reliable data collection and transmission. Performance is evaluated in terms of data collection accuracy, power consumption, and the efficiency of the solar panel/battery combination. Recommendations for improving system efficiency through optimization and integration of different sensors are also provided.**

I. Introduction

In recent years, the demand for energy efficiency and sustainable solutions has increased, promoting the development of smart systems based on renewable energy sources. This paper discusses the design and implementation of a solar-powered smart sensing system using the Raspberry Pi Pico microcontroller unit (MCU) with a dual-core Arm Cortex M0+ processor and the Amazon Web Services (AWS) IoT platform. The system will be powered by a solar panel and a 18650 lithium battery for nighttime use. Sensor data (e.g., from a DHT11 temperature and humidity sensor) will be collected periodically and analyzed using AWS services such as AWS Lambda and DynamoDB via the MQTT protocol through AWS IoT Core. Additionally, methods for operating the system in sleep mode to increase energy efficiency will be employed.

A. Background

Environmental data play an important role in various fields such as agriculture, meteorology, and ecology. However, the lack of reliable power sources in remote areas can make collecting and transmitting environmental data challenging. The decreasing cost of solar panels and the advantages of batteries demonstrate that solar energy is an ideal power source for environmental data collection systems. In regions like Turkey, where solar energy can be extensively utilized, the use of such systems is crucial.

II. System

A. Materials

The components used in this application include a 6V 250mA solar panel, a 3.7V 2200mAh 18650 lithium battery, Raspberry Pi Pico W, a temperature and humidity sensor (DHT11), a TP4056 charging module, and 2 Schottky diodes. Each component is selected based on factors such as cost, efficiency, and compatibility. The advantages and disadvantages of the battery protection provided by the TP4056 charging module (e.g., overcharge and discharge protection) will be explained.

Thonny Integrated Development Environment is used for coding, and lightweight codes written in MicroPython will be uploaded to the system. Data collection, transmission, and analysis processes will be carried out using the necessary libraries.

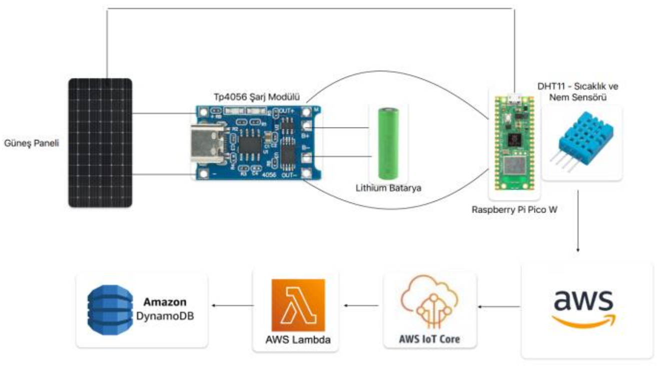


Figure 1: Architecture of the system.

B. Solar Panel

Two monocrystalline solar panels, each measuring 10x20 cm, have been used for the system. Monocrystalline solar panels, made from silicon, the second most abundant element on earth, offer high efficiency, durability, and relatively low cost. Approximately 80% efficiency was achieved from a panel providing 150mA of energy. [1]

C. Battery

The battery stores electrical energy from the panel as chemical energy. Lithium-ion batteries, commonly used in electronic devices, have been used here. While slightly more expensive than lead batteries, lithium-ion batteries are necessary for the durability required in our project. [2]

D. *Microcontroller*

The device that will transmit data from the sensor to the Cloud is the Raspberry Pi Pico W from the Raspberry Pi Foundation. Its most attractive feature is that it is readily available at reasonable prices, unlike its microcontroller cousins facing supply shortages. After the first version was released without a Wi-Fi module in 2021, the version with Wi-Fi and Bluetooth was introduced. Software support for Bluetooth came only in February 2023. This microcontroller, with its dual-core Arm Cortex M0+ architecture, is ideal for low-power applications. It has sleep and deep sleep modes that will save energy during non-data transmission cycles. With 2MB of internal flash memory and a 133 MHz clock frequency, it can comfortably handle our project. [3]

E. *DHT11*

The temperature and humidity sensor used will be the DHT11, which will take measurements at 10-minute intervals. Although it is less performant compared to the DHT22, it is quite suitable for our project due to its reasonable price and operating range between 3V and 5.5V. [4]

E. *Tp4056*

The TP4056 is a type of linear charging IC for single-cell lithium-ion batteries. It has a programmable charge current of up to 1A and a preset charge voltage of 4.2V. It also has two status outputs (led1, led2) to indicate that charging is ongoing or complete. The TP4056, despite protecting against discharge and high charge voltage, will stop charging when the battery reaches 4.2V. However, since our system will run continuously, this situation will wear out the battery. Therefore, an extra path will be added between the solar panel and the microcontroller to divert excess energy directly to the mini computer instead of loading it onto the battery.

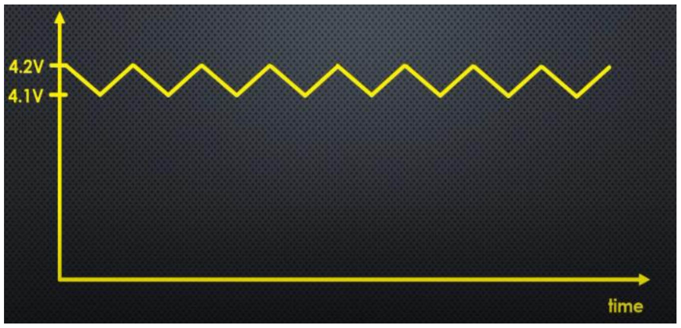


Figure 2: Charging cycle of TP4056.

E. *Schottky Diode*

These semiconductor combination diodes, known for their low forward voltage drop and fast switching capabilities, are used in rectification, signal regulation, switching, logic gates, and various applications like solar panels. In our project, their main contribution is to act as a barrier between the solar panel and Pico, protecting the panel and adding a diode to the positive terminal of the battery. This way, the source providing the highest voltage will keep the system running. If the panel does not generate energy at night, the battery will take over as the voltage drops below the battery's level.

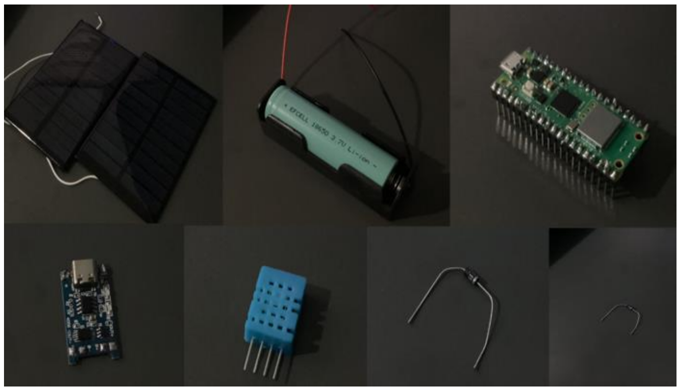
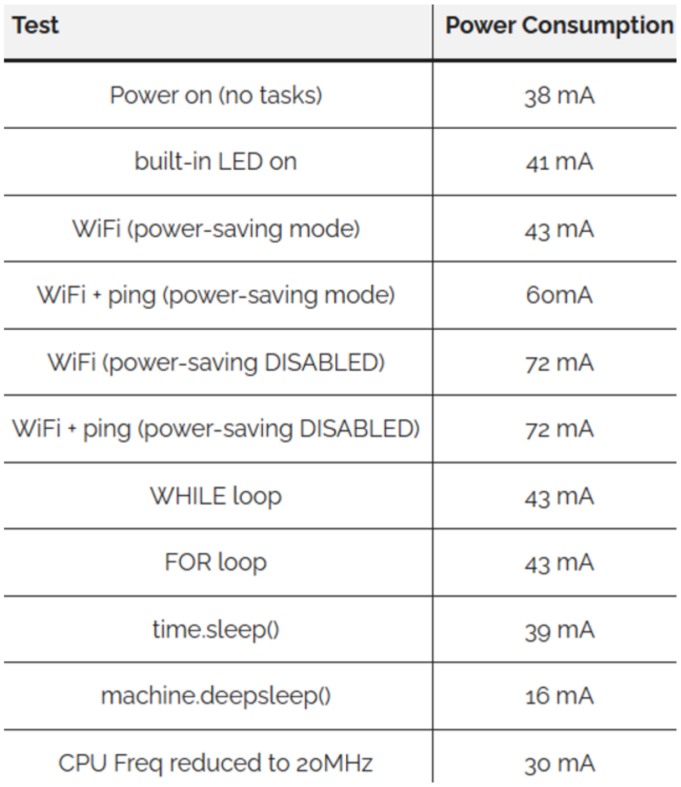


Figure 3: The materials presented in the first section.

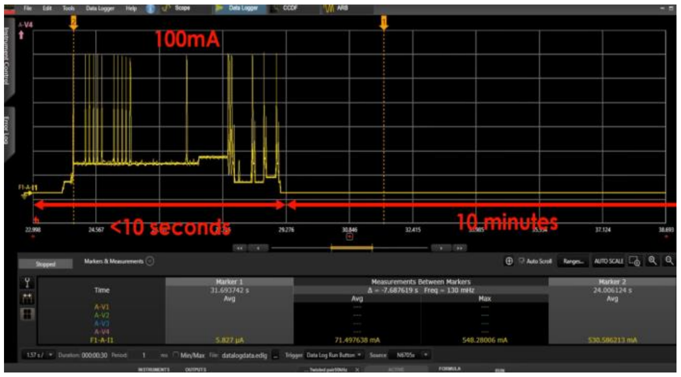
III. Installation

First, we prepare the Raspberry Pi Pico microcontroller unit and connect it to the computer for coding. The TP4056 charging module's IN positive and negative terminals are soldered. The module's output is connected to the positive (+) terminal of the lithium battery, and the energy from the solar panel is connected to the Schottky diodes. The DHT11 temperature and humidity sensor is connected to the 28 GPIO pins of the Raspberry Pi. Using MicroPython, we write the code that will read the sensor data and transmit it to AWS IoT Core via the MQTT protocol. The code will include methods to enable sleep mode (machine.deepsleep) for energy saving. After uploading the prepared code to the Raspberry Pi Pico, we obtain certificates from AWS and decode these PEM format certificates using the read_pem() function in Thonny. After creating a "Thing" in the AWS IoT Core account, we use AWS Lambda and DynamoDB services to analyze and store the incoming data. The system is tested to ensure that the data is collected, transmitted, and analyzed correctly.



Test	Power Consumption
Power on (no tasks)	38 mA
built-in LED on	41 mA
WiFi (power-saving mode)	43 mA
WiFi + ping (power-saving mode)	60mA
WiFi (power-saving DISABLED)	72 mA
WiFi + ping (power-saving DISABLED)	72 mA
WHILE loop	43 mA
FOR loop	43 mA
time.sleep()	39 mA
machine.deepsleep()	16 mA
CPU Freq reduced to 20MHz	30 mA

Figure 4: Consumption of Raspberry Pi Pico W according to different situations.



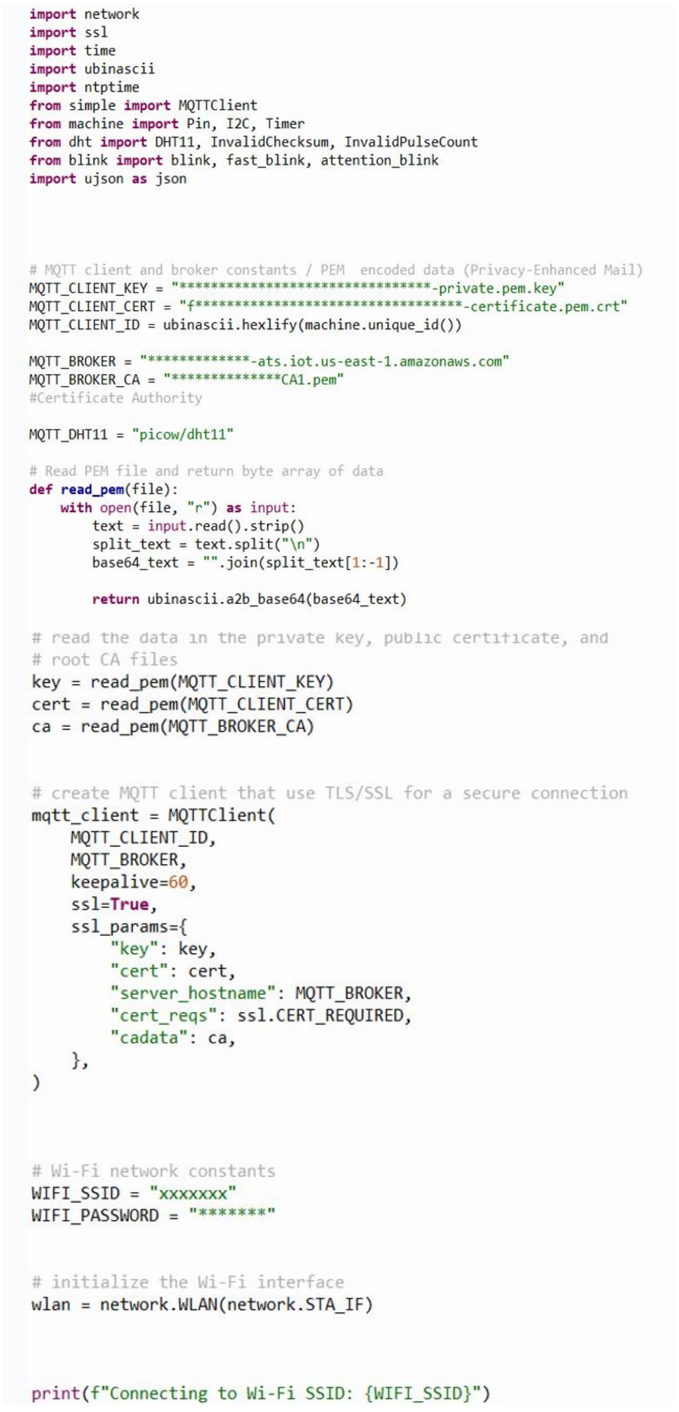
Time	Marker 1 31.693742 s Avg	Measurements Between Markers Δ = -7.687619 s Freq = 130 mHz Avg	Max	Marker 2 24.006124 s Avg
A-V1		---	---	
A-V2		---	---	
A-V3		---	---	
A-V4		---	---	
F1-A-I1	5.827 µA	71.497638 mA	548.28006 mA	530.586213 mA

Figure 5: Consumption of a 10-minute cycle.

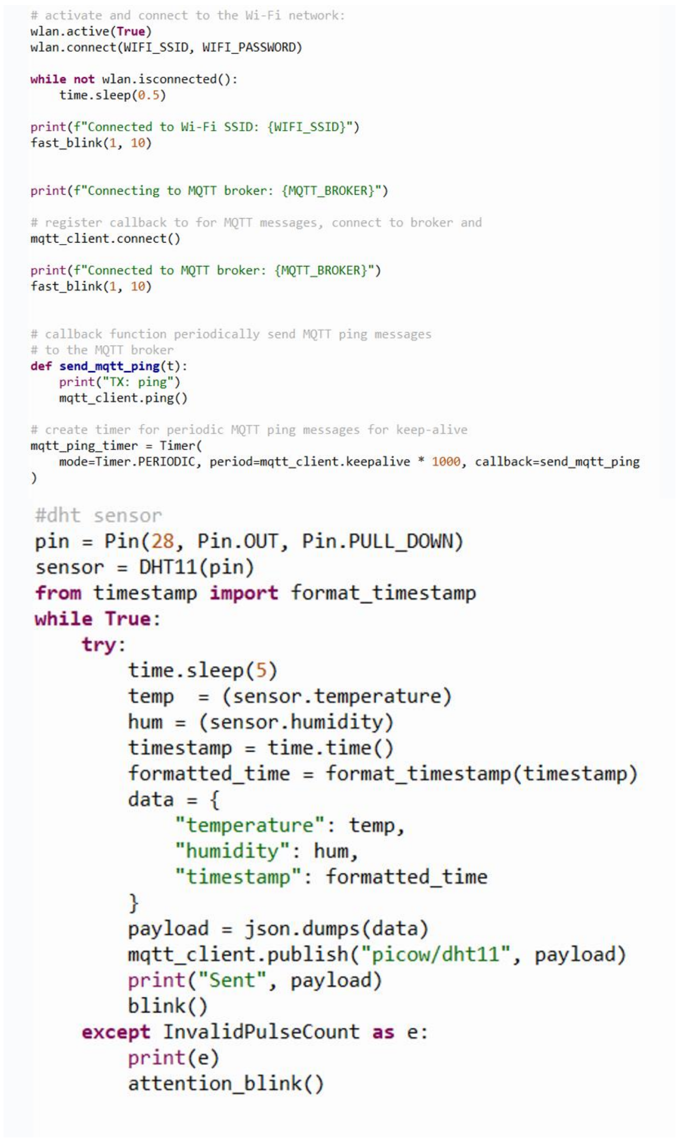
V. Recommendations

To achieve greater efficiency, optimizations such as sizing the solar panel/battery or integrating different sensors can be made. Additionally, considering the system features and usage scenarios, better solutions can be developed. For example, Schottky diodes, by their nature, also act as resistors, causing a voltage drop of approximately 0.3V. For this, a P-channel MOSFET with a low Gate Threshold could be used.

This article provides an explanation of the design and implementation of solar-powered smart sensing systems. More extensive methods can be used to evaluate system performance, energy efficiency, and increase the use of solar energy.



```
import network
import ssl
import time
import ubinascii
import ntptime
from simple import MQTTClient
from machine import Pin, I2C, Timer
from dht import DHT11, InvalidChecksum, InvalidPulseCount
from blink import blink, fast_blink, attention_blink
import ujson as json



# MQTT client and broker constants / PEM  encoded data (Privacy-Enhanced Mail)
MQTT_CLIENT_KEY = "*******************************-private.pem.key"
MQTT_CLIENT_CERT = "f*********************************-certificate.pem.crt"
MQTT_CLIENT_ID = ubinascii.hexlify(machine.unique_id())

MQTT_BROKER = "*************-ats.iot.us-east-1.amazonaws.com"
MQTT_BROKER_CA = "**************CA1.pem"
#Certificate Authority

MQTT_DHT11 = "picow/dht11"

# Read PEM file and return byte array of data
def read_pem(file):
    with open(file, "r") as input:
        text = input.read().strip()
        split_text = text.split("\n")
        base64_text = "".join(split_text[1:-1])

        return ubinascii.a2b_base64(base64_text)

# read the data in the private key, public certificate, and
# root CA files
key = read_pem(MQTT_CLIENT_KEY)
cert = read_pem(MQTT_CLIENT_CERT)
ca = read_pem(MQTT_BROKER_CA)


# create MQTT client that use TLS/SSL for a secure connection
mqtt_client = MQTTClient(
    MQTT_CLIENT_ID,
    MQTT_BROKER,
    keepalive=60,
    ssl=True,
    ssl_params={
        "key": key,
        "cert": cert,
        "server_hostname": MQTT_BROKER,
        "cert_reqs": ssl.CERT_REQUIRED,
        "cadata": ca,
    },
)


# Wi-Fi network constants
WIFI_SSID = "xxxxxxx"
WIFI_PASSWORD = "*******"


# initialize the Wi-Fi interface
wlan = network.WLAN(network.STA_IF)



print(f"Connecting to Wi-Fi SSID: {WIFI_SSID}")
```



```
# activate and connect to the Wi-Fi network:
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)

while not wlan.isconnected():
    time.sleep(0.5)

print(f"Connected to Wi-Fi SSID: {WIFI_SSID}")
fast_blink(1, 10)


print(f"Connecting to MQTT broker: {MQTT_BROKER}")

# register callback to for MQTT messages, connect to broker and
mqtt_client.connect()

print(f"Connected to MQTT broker: {MQTT_BROKER}")
fast_blink(1, 10)


# callback function periodically send MQTT ping messages
# to the MQTT broker
def send_mqtt_ping(t):
    print("TX: ping")
    mqtt_client.ping()

# create timer for periodic MQTT ping messages for keep-alive
mqtt_ping_timer = Timer(
    mode=Timer.PERIODIC, period=mqtt_client.keepalive * 1000, callback=send_mqtt_ping
)
#dht sensor
pin = Pin(28, Pin.OUT, Pin.PULL_DOWN)
sensor = DHT11(pin)
from timestamp import format_timestamp
while True:
    try:
        time.sleep(5)
        temp  = (sensor.temperature)
        hum = (sensor.humidity)
        timestamp = time.time()
        formatted_time = format_timestamp(timestamp)
        data = {
            "temperature": temp,
            "humidity": hum,
            "timestamp": formatted_time
        }
        payload = json.dumps(data)
        mqtt_client.publish("picow/dht11", payload)
        print("Sent", payload)
        blink()
    except InvalidPulseCount as e:
        print(e)
        attention_blink()
```

Figure 6: MicroPython source code.

References

[1] Priscila Gonçalves Vasconcelos Sampaio, Mario Orestes Aguirre González, Photovoltaic solar energy: Conceptual framework, Renewable and Sustainable Energy Reviews, Volume 74, 2017, Pages 590-601, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2017.02.081.

[2] Ghassan Zubi, Rodolfo Dufo-López, Monica Carvalho, Guzay Pasaoglu, The lithium-ion battery: State of the art and future perspectives, Renewable and Sustainable Energy Reviews, Volume 89, 2018, Pages 292-308, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2018.03.00

[3] https://developer.arm.com/Processors/Cortex-M0-PlusJ.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.*, in press.

[4] https://www.alldatasheet.com/datasheetpdf/pdf/1132088/ETC2/DHT11.html

[5] https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf