Design and Implementation of an IoT System: A Case Study in Remote Data Acquisition

Dr. M. Chandrashekar[1], M. Nandini[2], M. Nikhil Swami[3]

[1]*Professor, Dean, R&D ISL College of Engineering, Hyderabad, Telangana, 500005, India*

[2]*Embedded Systems R&D Engineer, Apollo Micro Systems Limited, Hyderabad- 500 076, Telangana, India.*

[3]*AI Systems R&D Engineer, Trip Gain Travel and Expense Management Solutions Pvt Ltd Bengaluru, Karnataka 560043, India*

***Abstract*—This paper presents the design and implementation of a bidirectional Bluetooth communication system using a Raspberry Pi Pico interfaced with an HC-05 Bluetooth module. The system allows remote control of an LED and the acquisition of analog sensor data via the Pico's ADC. Commands received wirelessly from a mobile device or via a USB-connected laptop (running Thonny) are processed by the Pico, which then sends control signals or sensor data back through the HC-05. Both development (with Thonny) and standalone modes are addressed. Experimental results demonstrate reliable LED control, accurate ADC readings, and system responsiveness under varying conditions. This implementation is useful in IoT applications such as remote monitoring, industrial automation, and smart home solutions.**

***Index Terms*—Bluetooth communication, Raspberry Pi Pico, HC-05, ADC, embedded systems, MicroPython, Internet of Things (IoT).**

I. INTRODUCTION

Wireless communication is a cornerstone of modern embedded systems and IoT applications. This work focuses on integrating bidirectional Bluetooth communication with sensor data acquisition on a cost-effective microcontroller platform. The Raspberry Pi Pico, a low-cost, high-performance board based on the RP2040 chip, is paired with an HC-05 Bluetooth module to establish a communication link with mobile devices. The system is designed to control an LED and acquire analog data from a sensor using the onboard ADC. The solution is developed in MicroPython and is demonstrated in both a development environment using Thonny and as a standalone device.

II. SYSTEM ARCHITECTURE

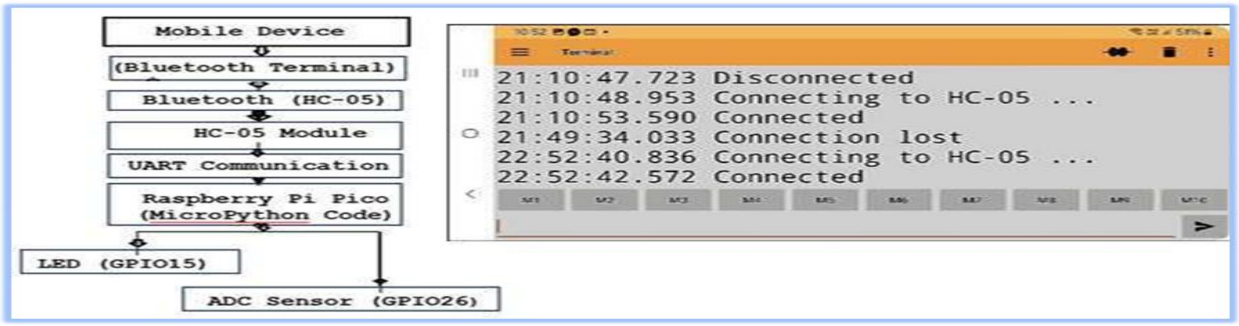
The system architecture consists of three primary components:

1. Raspberry Pi Pico: Executes the control and data acquisition code.
2. HC-05 Bluetooth Module: Facilitates wireless communication with mobile devices.
3. Peripheral Devices: An LED (for visual feedback) and an analog sensor interfaced to the Pico's ADC.

A. Power Supply

For development, the Pico is powered via USB through a laptop. For standalone operation, an external USB power adapter or power bank is used.

B. Block Diagram



```
21:10:47.723 Disconnected
21:10:48.953 Connecting to HC-05 ...
21:10:53.590 Connected
21:49:34.033 Connection lost
22:52:40.836 Connecting to HC-05 ...
22:52:42.572 Connected
```

Fig. 1: Data Flow Diagram

C. To pair mobile phone with the HC-05 Bluetooth module.

Download Serial blue tooth Terminal. First power on the HC-05; its LED blink rapidly, indicating it's in pairing mode. On the mobile device, enable Bluetooth and search for available devices; select "HC-05" and, when prompted, enter the default pairing code, typically "1234" or "0000". Once paired, install a Bluetooth terminal application, such as "Serial Bluetooth Terminal" for Android. Open the app, connect to the HC-05, and you'll be able to send and receive data wirelessly between your mobile phone and the module

III. HARDWARE IMPLEMENTATION

A. Components and Wiring

The following components are used:

- Raspberry Pi Pico
- HC-05 Bluetooth Module
- LED with a current-limiting resistor (e.g., 220 f2)
- Analog Sensor/Potentiometer
- Jumper Wires and Breadboard

Wiring Details:

- HC-05 Module:

0 VCC: Pico's 3.3V (or 5V if supported)

0 GND: Common ground

0 TX: Pico's RX (GPIOI)

0 RX: Pico's TX (GPIOO) via a voltage divider (10 kf2 and 20 kfl resistors)

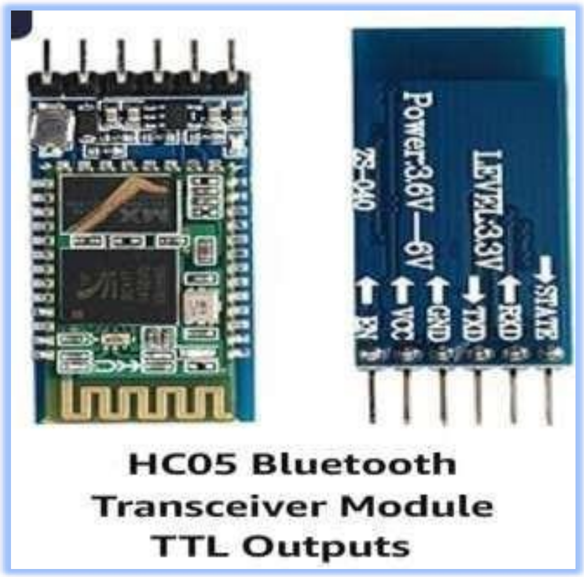


Fig. 2: HC-05 Bluetooth Module

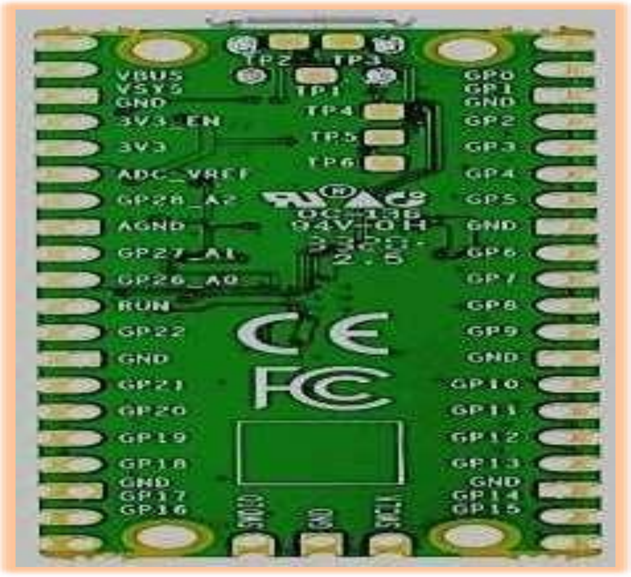


Fig. 3: Raspberry Pi Pico FO Pins

- LED:

0 Anode: Connected to GPIO15 through a resistor

0 Cathode: Ground

- ADC Sensor:

0 Signal output: Connected to ADC input (GPIO26)

0 Power and ground as per sensor specifications

IV. SOFTWARE DESIGN AND IMPLEMENTATION

The MicroPython code performs three main functions:

1. Command Reception: Processes commands received via the HC-05 from a mobile device.
2. LED Control: Toggles an LED on GPIO15 based on commands (“1” to turn on, “0” to turn off).
3. ADC Reading**:** When the command “ADC” is received, the Pico reads the analog value from GPIO26, computes the corresponding voltage, and sends this data back via Bluetooth.

Key Implementation Aspects:

- UART Communication: Configured at 9600 baud for communication with the HC-05.
- Non-Blocking Shell Input: Utilizes the select module to check for input from the Thonny shell without blocking execution.
- Command Handling: Supports LED control,

ADC reading, and general message transmission.

V. EXPERIMENTAL SETUP AND RESULTS

A. Operating Procedure

1. Development Phase:

0 The Pico is connected to a laptop via USB.
O Thonny IDE is used to upload and debug the code.
O Commands are sent from a mobile device and Thonny shell. O Observations include LED toggling and ADC value readings.

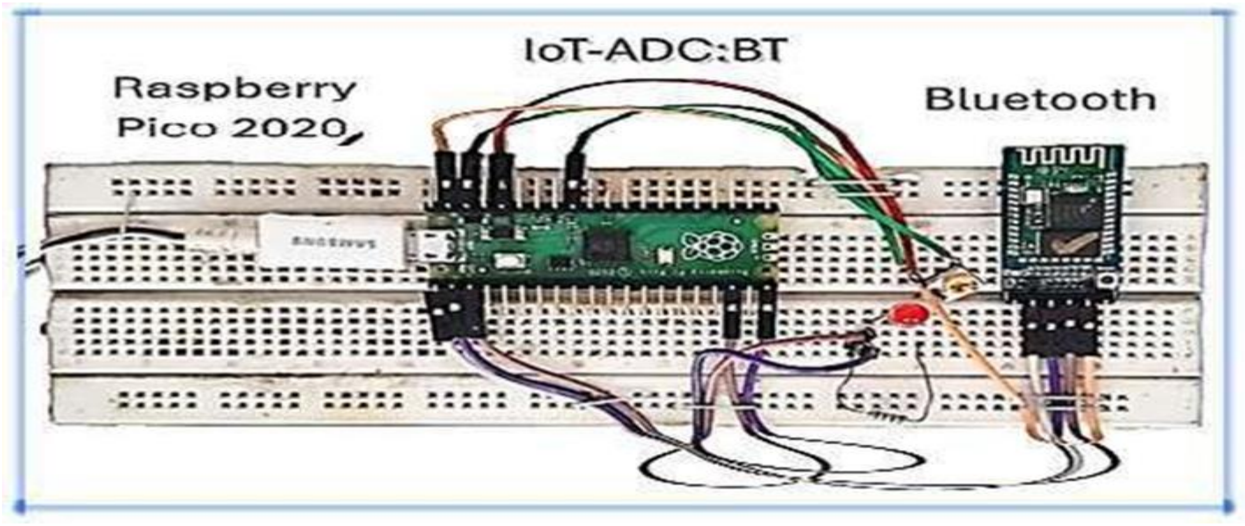


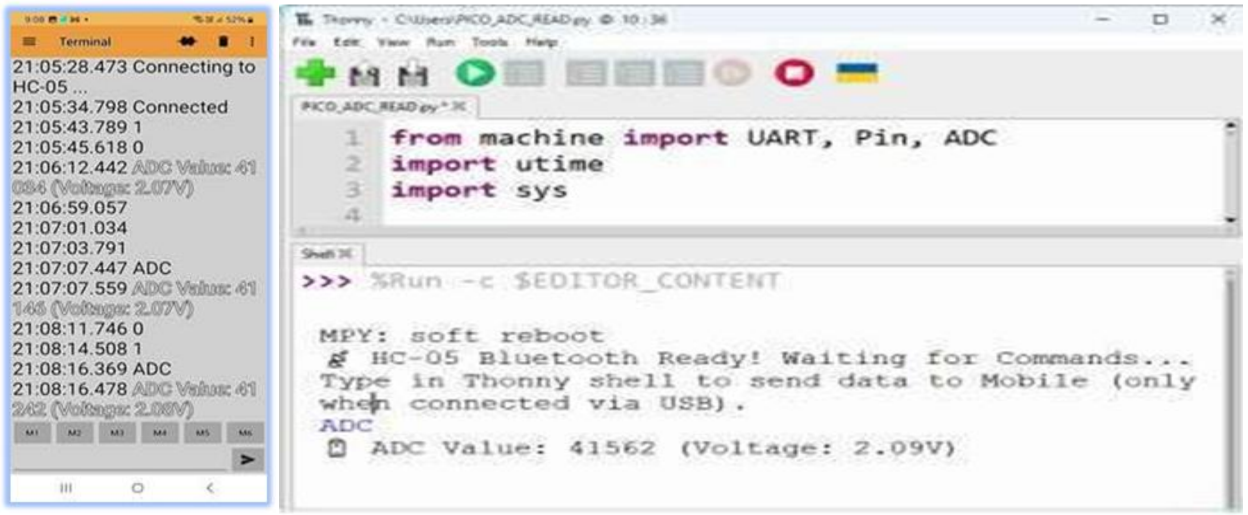
Fig. 4: IoT Prototype

2. Standalone Operation:

0 The Pico is powered externally.
0 The HC-05 module enters pairing mode.
0 A mobile device connects via a Bluetooth terminal application.
0 Commands (“1”, “0”, and “ADC”) are processed successfully.

B. Results and Analysis

- LED Control: Commands “1” and “0” reliably toggle the LED.
- ADC Measurement: Readings accurately reflect the sensor's output within a 3.3V reference.
- Bidirectional Communication: The system maintains responsiveness in all modes.
- Potential Limitations: Bluetooth interference may introduce latency in noisy environments.



```
21:05:28.473 Connecting to HC-05 ...
21:05:34.798 Connected
21:05:43.789 1
21:05:45.618 0
21:06:12.442 ADC Value: 41084 (Voltage: 2.07V)
21:06:59.057
21:07:01.034
21:07:03.791
21:07:07.447 ADC
21:07:07.559 ADC Value: 41146 (Voltage: 2.07V)
21:08:11.746 0
21:08:14.508 1
21:08:16.369 ADC
21:08:16.478 ADC Value: 41242 (Voltage: 2.08V)
```

```
from machine import UART, Pin, ADC
import utime
import sys
```

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
  HC-05 Bluetooth Ready! Waiting for Commands...
 Type in Thonny shell to send data to Mobile (only
 when connected via USB).
 ADC
  ADC Value: 41562 (Voltage: 2.09V)
```

Fig. 5: ADC Readings on Mobile App and Thonny Shall

VI. CONCLUSION

This paper presents a robust solution for bidirectional Bluetooth communication and sensor data acquisition using a Raspberry Pi Pico and an HC-05 module. The system effectively controls an LED and reports ADC sensor data based on wireless commands from a mobile device or a laptop. Future work may focus on expanding command functionality, integrating additional sensors, and improving error handling.

REFERENCES

[1] Raspberry Pi Pico Documentation. [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/
[2] MicroPython Documentation. [Online]. Available: https://docs.micropython.org/en/latest/
[3] HC-05 Bluetooth Module Datasheet and Tutorials.
[4] Research papers on embedded systems and IoT communication protocols.