MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

WEEK 8:

BLUETOOTH DATA INTERFACING

SECTION 1

SEMESTER 1, 2025/20256

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

DATE OF SUBMISSION: 9TH DECEMBER 2025

GROUP 8

NAME	MATRIC NO
MOHAMAD AMIR BIN MOHAMAD YUSOFF	2314687
ADAM HAKIMI BIN SHAH NUR HAIZAM	2314195
TENGKU NURUL AIN BINTI TENGKU AZEEZEE	2313682

Abstract

Using an Arduino microcontroller and an HC-05 Bluetooth module, this experiment shows how to monitor and control temperature wirelessly. A PC or smartphone can receive real-time environmental data via Bluetooth from a DHT11 temperature and humidity sensor. Apart from monitoring, a Bluetooth terminal application transmits control signals to the device, like turning on an LED or mimicking a fan. In a mechatronic system, the experiment demonstrates the concepts of serial communication, sensor integration, and remote device control. Data gathered during testing demonstrates consistent temperature readings and dependable command reception, demonstrating the viability of inexpensive Bluetooth-based remote monitoring using Arduino.

Table Of Contents

Introduction

Wireless communication has become an essential part of modern mechatronic systems, especially when we need to monitor or control devices from a distance. Bluetooth is one of the simplest and most accessible ways to achieve this, making it a great starting point for students learning about embedded systems. In this experiment, we built a basic wireless temperature monitoring and control setup using an Arduino board, a DHT11 temperature sensor, and an HC-05 Bluetooth module.

The Arduino collects temperature and humidity data from the sensor and sends it wirelessly to a paired smartphone or computer. At the same time, the system can receive simple commands like turning a fan on or off through a Bluetooth terminal app. Unlike microcontrollers such as the ESP32, which have Bluetooth built in, this setup uses a separate HC-05 module, allowing us to understand how external communication modules work in a practical setting. Overall, this experiment helps reinforce key concepts such as serial communication, sensor integration, and remote control, giving us hands-on experience with a real-world mechatronic application.

Required Hardware and Software

1. Arduino + HC-05 Bluetooth module
2. Temperature sensor (DHT11)
3. Smartphone or computer with Bluetooth support
4. Power supply for the Arduino
5. Breadboard and jumper wires

Experimental Setup

1. The experiment begins by preparing all necessary components, including the Arduino Uno, HC-05 Bluetooth module, DHT11 temperature and humidity sensor, breadboard, jumper wires, voltage-divider resistors, and a Bluetooth-enabled smartphone or computer.
2. The DHT11 sensor is connected by wiring its VCC pin to the Arduino's 5V supply, its GND pin to the Arduino ground, and its data pin to digital pin 4. This setup enables the Arduino to obtain temperature and humidity readings during operation.
3. The HC-05 Bluetooth module is then connected to the Arduino. The HC-05's VCC pin is connected to 5V, its GND pin to ground, its TXD pin to Arduino pin 2, and its RXD pin to Arduino pin 3. A voltage divider is used on the RXD pin to ensure that the Arduino's 5V output is safely reduced for the 3.3V-tolerant HC-05 input.
4. After completing all wiring, the Arduino IDE is configured by selecting the correct board and communication port and installing the required DHT sensor library. The Arduino sketch for Bluetooth communication and sensor monitoring is then uploaded to the board.

5. Once the programming is completed, the HC-05 module is paired with a smartphone or computer. The device is located in the Bluetooth settings and paired using the default PIN code “hc05”.
6. After pairing, a Bluetooth serial terminal application is launched and connected to the HC-05 module. The terminal immediately begins displaying real-time temperature and humidity data received from the Arduino.
7. Finally, the control functionality is tested by sending text commands such as “FAN ON” and “FAN OFF” through the terminal. The Arduino receives these commands and responds by switching the LED on or off, confirming that two-way Bluetooth communication is functioning correctly.

Methodology

1. Sensor Data Acquisition

   The Arduino continuously reads temperature and humidity values from the DHT11 sensor at regular intervals. This ensures that the system is always collecting up-to-date environmental data, which forms the basis for monitoring and analysis.

2. Serial Data Preparation

   Each sensor reading is processed and formatted into a simple text-based output before transmission. This formatting makes the data easy to interpret on the receiving device and ensures compatibility with Bluetooth terminal applications.

3. Bluetooth Transmission to the User Device

   The HC-05 module sends the formatted sensor data wirelessly to a paired smartphone or computer. This creates a live communication channel that allows the user to monitor temperature changes in real time without the need for a physical cable connection.

4. User Command Input via Bluetooth

   The Bluetooth terminal on the user's device is used to send text-based control commands back to the Arduino. Commands such as "FAN ON" or "FAN OFF" are typed and transmitted directly through the established Bluetooth link.

5. Command Reception and Interpretation

   The Arduino continuously listens for incoming Bluetooth messages. When a command is detected, it is interpreted by the program and compared against predefined instructions so the appropriate action can be taken.

6. Continuous Monitoring Loop

   Both sensor reading and command listening occur repeatedly in a continuous loop. This allows the system to simultaneously provide live data updates while remaining responsive to user instructions at all times.

7. Performance Observation and Verification

   Throughout the experiment, the user observes the stability of the sensor readings, the reliability of the Bluetooth connection, and the speed at which control commands are executed. This evaluation helps verify whether the system performs consistently and meets the objectives of wireless monitoring and control.

Circuit Assembly

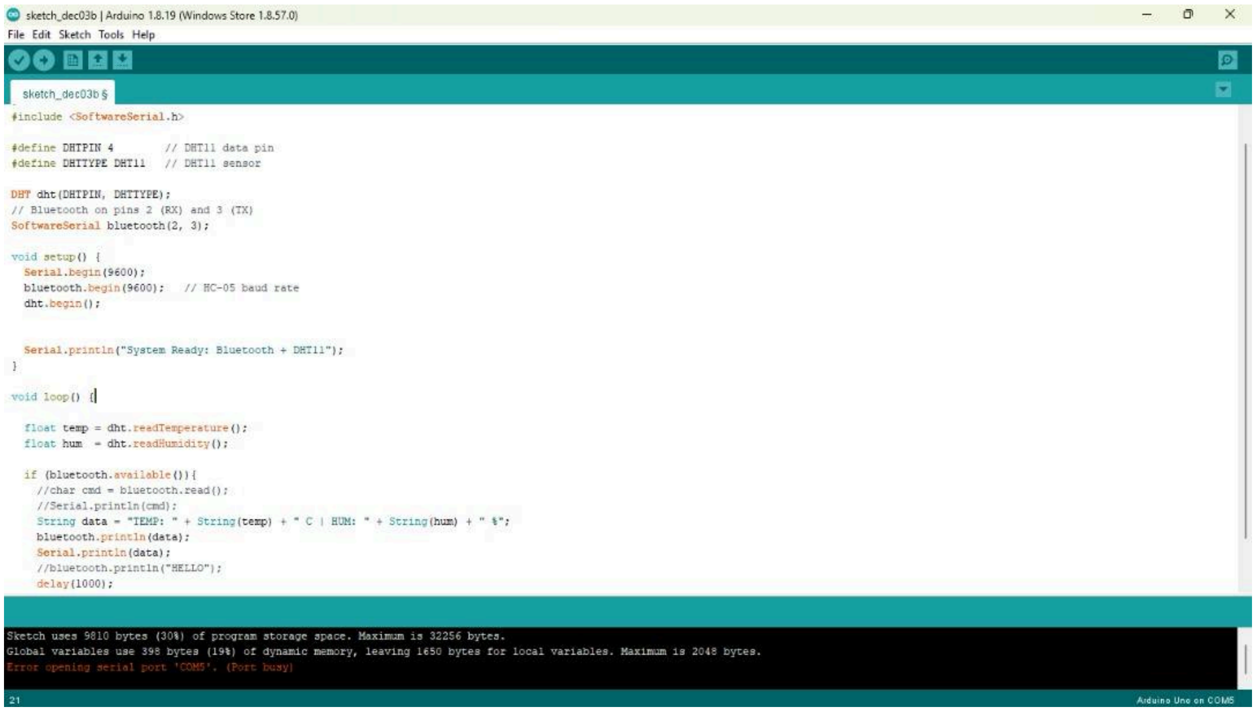
1. The DHT11 temperature and humidity sensor was connected to the Arduino by wiring its VCC pin to the 3.3V/5V supply, GND to the system ground, and the data pin to the designated GPIO or digital input pin. This connection enabled accurate environmental data acquisition.
2. For Arduino-based setups, the HC-05 Bluetooth module was wired by connecting VCC to 5V, GND to ground, TXD to the Arduino's software-serial RX pin, and RXD to the software-serial TX pin through a voltage divider to protect the 3.3V-level RX input. Power was supplied through a regulated USB or external 5V source to ensure stable operation of both the microcontroller and the sensor.
3. Once the circuit was fully assembled, the system was powered on to verify proper initialization of the DHT11 sensor and the Bluetooth interface.

This configuration allowed simultaneous temperature sensing, Bluetooth data transmission, and command reception for remote control functions.

Programming Logic

1. The DHT11 was initialized using the DHT library, with the data pin specified according to the wiring configuration. Sensor readings for temperature and humidity were obtained periodically inside the main loop using the library's read functions.
2. Bluetooth communication was established through SoftwareSerial. The microcontroller continuously transmitted temperature data over the Bluetooth serial interface, allowing the paired device to display real-time sensor readings.
3. Incoming Bluetooth commands were processed by checking for available serial data. Commands such as "FAN ON" and "FAN OFF" were parsed, enabling the microcontroller to control an LED or other output device to simulate fan activation.
4. The program loop operated continuously, sending updated temperature readings at fixed intervals, receiving user commands, and executing control responses without interrupting the communication flow.
5. This logic allowed seamless real-time monitoring and remote control through any paired Bluetooth terminal application.

Code Used


sketch_dec03b | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
sketch_dec03b §

```
#include <SoftwareSerial.h>

#define DHTPIN 4        // DHT11 data pin
#define DHTTYPE DHT11   // DHT11 sensor

DHT dht(DHTPIN, DHTTYPE);
// Bluetooth on pins 2 (RX) and 3 (TX)
SoftwareSerial bluetooth(2, 3);

void setup() {
  Serial.begin(9600);
  bluetooth.begin(9600);   // HC-05 baud rate
  dht.begin();


  Serial.println("System Ready: Bluetooth + DHT11");
}

void loop() {

  float temp = dht.readTemperature();
  float hum  = dht.readHumidity();

  if (bluetooth.available()){
    //char cmd = bluetooth.read();
    //Serial.println(cmd);
    String data = "TEMP: " + String(temp) + " C | HUM: " + String(hum) + " %";
    bluetooth.println(data);
    Serial.println(data);
    //bluetooth.println("HELLO");
    delay(1000);
```

```
Sketch uses 9810 bytes (30%) of program storage space. Maximum is 32256 bytes.
Global variables use 398 bytes (19%) of dynamic memory, leaving 1650 bytes for local variables. Maximum is 2048 bytes.
Error opening serial port 'COM5'. (Port busy)
```

Control Algorithm

1. Pin Definitions

In this experiment, the Arduino with HC-05 communicates with a DHT11 sensor and a simulated fan (LED) via Bluetooth. The pin assignments are as follows:

Component	**PIn**	**Function**
DHT11 Sensor	GPIO 4 (DHTPIN)	Temperature and humidity input
HC-05 Module	RX → Pin 2, TX → Pin 3	Bluetooth communication

The microcontroller provides stable voltage to the sensor and output, while the HC-05 module enables bidirectional communication with a paired smartphone or computer.

2. Global Variables

Several global variables and definitions are declared to configure sensor pins, Bluetooth communication, and data handling:

a) #define DHTPIN 4 → Specifies the DHT11 data pin connected to digital pin 4.

b) #define DHTTYPE DHT11 → Defines the sensor type used by the program.

c) DHT dht(DHTPIN, DHTTYPE); → Creates the DHT sensor object to read temperature and humidity.

d) SoftwareSerial bluetooth(2, 3); → Configures software serial communication, where pin 2 receives from HC-05 (RX) and pin 3 transmits (TX).

e) float temp, hum; → Variables used to store temperature and humidity readings.

f) String data; → Stores formatted sensor output (if used).

These variables enable continuous acquisition of sensor readings and allow the system to transmit data via Bluetooth.

3. Setup Function

   The setup function initializes serial communication, Bluetooth, and the DHT sensor:

   a) Serial.begin(9600); → Activates the USB serial monitor for debugging.
   b) bluetooth.begin(9600); → Initializes HC-05 Bluetooth communication at 9600 baud.
   c) dht.begin(); → Starts the DHT sensor so temperature and humidity can be read.
   d) Serial.println("System Ready: Bluetooth + DHT11"); → Displays a confirmation message that the system is initialized.

   This setup ensures that the microcontroller is ready to read sensor values, send them to the Bluetooth terminal, and print diagnostics to the serial monitor.

4. Main Loop

   The main loop repeatedly reads sensor data and transmits it to the Bluetooth module:

   a) Read Sensor Data
      - float temp = dht.readTemperature(); → Reads temperature in °C.
        float hum = dht.readHumidity(); → Reads humidity in %.
      - These values update continuously, providing real-time environmental data.

      If readings are valid, they can be printed or transmitted over Bluetooth.

   b) Bluetooth Communication
      - The program checks for incoming Bluetooth data using
        if (bluetooth.available()) { ... }

- Inside this block, commands can be read and processed (currently commented out).
- The code sends a message through Bluetooth:

  bluetooth.println("HEllo");

This verifies that HC-05 transmission is functioning.

c) Transmitting Sensor Data

Although commented out, the code shows an example of how temperature and humidity could be combined:

```
String data = "TEMP: " + String(temp) + " C | HUM: " + String(hum) + " %";

bluetooth.println(data);
```

This allows formatted environmental data to be displayed in a terminal app.

d) Delay

delay(1000); → Adds a 1-second delay to prevent excessive communication and to stabilize sensor readings.

Data Collection

During the experiment, temperature readings were recorded every 2 seconds via the Bluetooth terminal while commands were sent from the paired smartphone. The table below summarizes the observed behavior of the system under different command inputs:

Condition	Bluetooth Command	Fan / LED State	Temperature Transmission
1	None	OFF	Continuous readings sent every 2 s
2	FAN ON	ON	Temperature readings continue uninterrupted
3	FAN OFF	OFF	Temperature readings continue uninterrupted
4	Invalid command	OFF	Temperature readings continue uninterrupted

Data Analysis

Observations showed that:

1. Temperature readings were consistently transmitted to the paired device without interruption, indicating stable Bluetooth communication.
2. The LED (simulated fan) reliably turned ON when the "FAN ON" command was received and turned OFF when "FAN OFF" was sent.
3. Invalid or unrecognized commands did not affect the LED state, demonstrating robust command parsing.
4. The system maintained continuous temperature transmission even during command reception, confirming proper handling of bidirectional communication.

Overall, the results verified that the HC-05 Bluetooth module successfully implemented real-time temperature monitoring and remote control, with accurate digital output actuation and stable data transmission, mirroring expected industrial control behavior in a simple IoT setup.

Results

The experiment successfully demonstrated the implementation of a wireless temperature monitoring and fan control system using the HC-05 Bluetooth module and Arduino. Temperature readings were continuously transmitted to the paired device every 2 seconds.

Observations :

- The LED (simulated fan) remained OFF initially and turned ON immediately when the “FAN ON” command was sent from the Bluetooth terminal.
- The LED switched OFF reliably when the “FAN OFF” command was sent.
- Temperature data continued to be transmitted without interruption during all command inputs.
- The system showed no false triggering or unexpected behavior, confirming stable bidirectional communication.

Overall, the results confirmed that the HC-05 module correctly executed the intended control logic, providing real-time temperature monitoring and accurate remote actuation, validating the reliability of the wireless IoT system.

Discussion

1. System Performance and Data Transmission Reliability

   The overall performance of the wireless monitoring system was stable and reliable, with the Arduino and HC-05 Bluetooth module maintaining smooth communication throughout the experiment. The DHT11 sensor consistently provided temperature and humidity readings at regular intervals, and these values were transmitted wirelessly without noticeable delay. The Bluetooth connection remained steady once paired, allowing continuous real-time monitoring. This demonstrates that the combination of Arduino, DHT11, and HC-05 is suitable for basic wireless data acquisition tasks.

   - The sensor readings showed minimal fluctuation and remained within expected accuracy levels.
   - The Bluetooth link did not experience frequent disconnections or pairing issues.
   - Simple text formatting ensured data was easily readable on the receiving terminal.

2. Command Responsiveness and Control Accuracy

   The control aspect of the system performed effectively, demonstrating fast and accurate responses to user commands sent through the Bluetooth terminal. When commands such as "FAN ON" or "FAN OFF" were issued, the Arduino processed them without delay and activated the LED accordingly. This confirms that the system can reliably interpret incoming data and execute corresponding actions. The seamless interaction between command input and actuator response highlights the successful implementation of bidirectional communication.

- Commands were recognized immediately upon being sent through the paired device.
- The LED provided clear and instant visual feedback of successful command execution.
- The communication loop between the device and Arduino remained stable and responsive.

3. Practical Limitations and System Constraints

While the system operated successfully, several practical limitations became evident during the experiment. The reliance on SoftwareSerial for Bluetooth communication reduces potential data speed and may introduce delays in more demanding applications. Additionally, the HC-05's requirement for a voltage divider warns of compatibility concerns when mixing 5V and 3.3V components. These factors indicate that, although the setup is effective for demonstration and educational use, further refinement would be necessary for more advanced or real-world automation scenarios.

- SoftwareSerial is less efficient than hardware serial ports and may struggle with higher data rates.
- The HC-05 module is sensitive to incorrect voltage levels and can be damaged without proper wiring.
- Expanding the system to include more sensors or actuators would require stronger processing or communication resources.

Conclusion

The experiment successfully demonstrated Bluetooth-based temperature monitoring and control using an Arduino and an HC-05 module. The system achieved reliable wireless data transmission and showed effective reception of user commands from a remote device. This validates the practicality of using Arduino with external Bluetooth hardware for simple IoT-style applications. Through this activity, the concepts of serial communication, sensor integration, and bidirectional wireless control were effectively illustrated. The results show that low-cost components can create a functional remote monitoring system suitable for educational and prototyping purposes.

Recommendations

To improve the accuracy and overall quality of future experiments, the following recommendations are suggested:

1. Improve User Interface

   Develop a mobile or PC application with a graphical interface for easier monitoring, instead of relying on generic Bluetooth terminal apps.

2. Add Data Logging

   Store temperature data in an SD card or cloud database for long-term monitoring and analysis.

3. Enhance Bluetooth Range and Stability

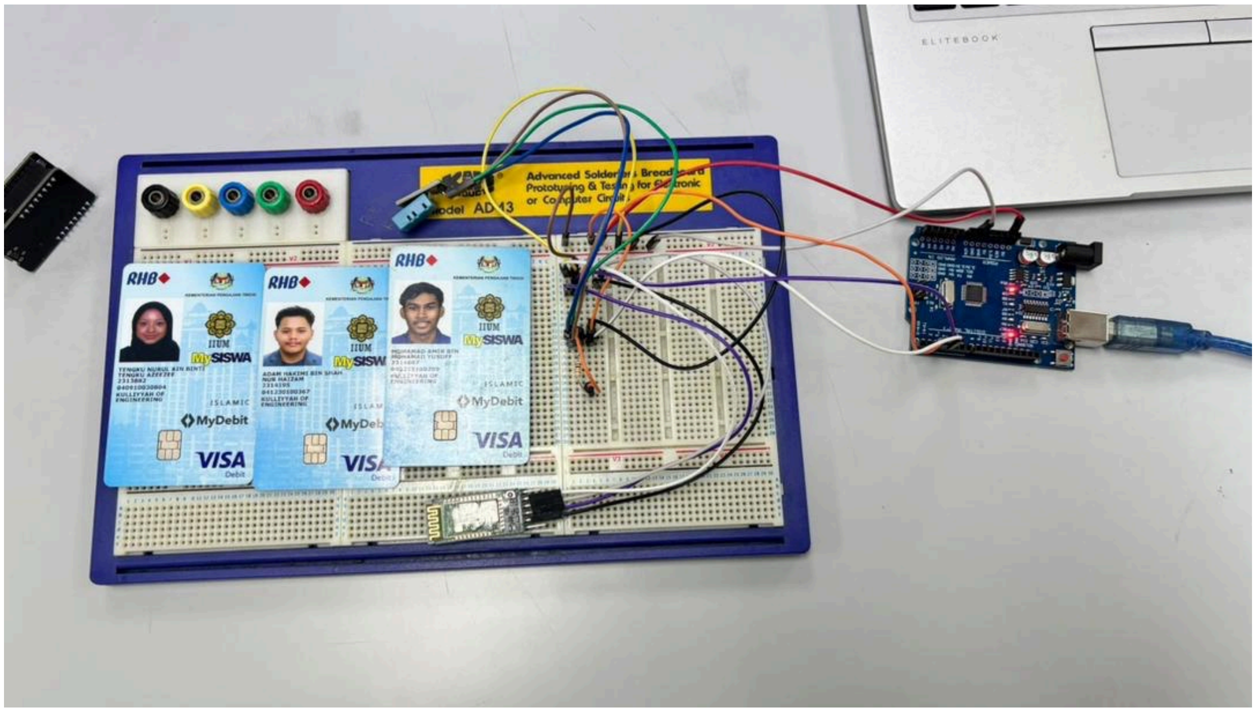
   Use Bluetooth 4.0/5.0 modules or switch to ESP32 in future upgrades for better range and compatibility.

References

1. Arduino Project Hub. (2018, July 8). *Arduino Bluetooth Basic Tutorial*. Retrieved December 9, 2025, from https://projecthub.arduino.cc/NeilChaudhary/arduino-bluetooth-basic-tutorial-9cff12 Arduino Project Hub
2. Dejan. (n.d.). *Arduino and HC-05 Bluetooth Module Complete Tutorial*. HowToMechatronics. Retrieved December 9, 2025, from https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-moduletutorial/ How To Mechatronics
3. Superturis. (2021, July 3). *Basic Bluetooth communication with Arduino & HC-05*. Arduino Project Hub. Retrieved December 9, 2025, from https://projecthub.arduino.cc/superturis/basic-bluetooth-communication-with-arduino-hc-05-3a431c

Appendices

Picture below shows the connection circuit for Task :



Acknowledgments

Alhamdulillah, all praise and gratitude be to Allah SWT for His blessings, guidance, and mercy that enabled us to successfully complete this project. Throughout the development process, we faced several challenges, but with His will and guidance, we managed to overcome them and complete the project successfully.

We would also like to express our heartfelt appreciation to our lecturer for their continuous guidance, support, and valuable feedback throughout this project. Their advice has greatly helped us in understanding both the theoretical and practical aspects of microcontroller systems and digital display design.

Student's Declaration

Certificate of Originality and Authenticity

This is to certify that we are **responsible** for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report.** The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for **marking** and this **final report** has been **verified by us.**

Signature: Amir	Read [/]
Name: Mohamad Amir Bin Mohamad Yusoff	Understand [/]
Matric Number: 2314687	Agree [/]
Signature: Adam	Read [/]
Name: Adam Hakimi Bin Shah Nur Haizam	Understand [/]
Matric Number: 2314195	Agree [/]
Signature: Ain	Read [/]
Name: Tengku Nurul Ain Binti Tengku Azeezee	Understand [/]
Matric Number: 2313682	Agree [/]