Final Project: Gardenhouse Control

Amanda Balderas Arias
Fernando Yañez García
Erick Abraham Galicia De La Rosa

1 Objective

The objective of this project is to implement an automated system for controlling a greenhouse using a Raspberry Pi and an Arduino Uno, which monitor and regulate critical environmental variables such as temperature, humidity, and irrigation. It includes a user interface, which is used through a web page, allowing users to monitor and manually or automatically adjust system parameters. The project also includes a historical data logging system, which will facilitate control and decision-making to improve the greenhouse's efficiency, always ensuring the environment is suitable for plant growth.

2 Materials List

- Arduino UNO Board
- 4B Raspberry Pi Board
- DHT11 Humidity Sensor
- LM35 Temperature Sensor
- Water Pump
- 12V Fan
- Soil Temperature Sensor
- Relay to control the water pump
- Incandescent Bulb
- 1N4007 Diodes
- 2N2222 Transistor
- Electrical Resistors
- TRIAC BT138
- MOC 3021 Optocoupler
- 4N25 Optocoupler
- Breadboards
- Connection Cables
- Fish Tank (or Aquarium)

3 Operation of the electronic components

3.1 Arduino UNO Board

Arduino is an open-source electronics platform that integrates accessible hardware and software for prototyping and development. Specifically, Arduino boards are designed to interpret various types of inputs, such as sensor readings, button presses, or data from online sources, and subsequently translate these into specific outputs, such as motor activation, LED illumination, or data publication on digital platforms. In this context, sensors play a pivotal role in providing the necessary input data for Arduino projects.

3.2 Raspberry Pi Board

The 4B Raspberry Pi is a low-cost, compact single-board computer designed for educational projects, prototyping, and automation. It features an ARM Cortex-A processor, up to 8 GB of RAM, USB ports, HDMI output, and Wi-Fi and Ethernet connectivity. Storage is handled via a microSD card, and it includes GPIO pins for interacting with external devices like sensors and actuators. It primarily runs Raspberry Pi OS, based on Linux, but also supports other operating systems. It's a versatile and accessible tool, ideal for electronics, robotics, and system control projects.

3.3 DHT11 Humidity Sensor

The DHT11 is a low-cost, low-power digital temperature and humidity sensor. It is designed to measure relative humidity in the range of 20% to 80% with a ±5% accuracy, and temperature in the range of 0°C to 50°C with a ±2°C accuracy. The sensor communicates data digitally, using a single pin for information transmission. It is widely used in electronics and automation projects, especially when a simple sensor is needed to measure basic environmental conditions. Its easy integration with microcontrollers like Arduino and Raspberry Pi makes it ideal for applications in climate monitoring and control systems.

3.4 LM35 Temperature Sensor

The LM35 is an integrated temperature sensor designed to measure temperatures in degrees Celsius with high accuracy and no need for external calibration. Its output is linearly proportional to the measured temperature, generating 10 mV per degree Celsius. It operates within a voltage range of 4 to 30 V DC, consumes less than 133 μA, and has a typical accuracy of 0.5 C at room temperature.

3.5 1N4007 Diode

The 1N4007 is a general-purpose rectifier diode known for its ability to handle high voltages and currents. It has a maximum reverse voltage of 1000 V and a maximum forward current of 1 A, making it suitable for alternating current (AC) rectification applications. Its forward voltage drop is approximately 0.7 V at nominal current, and although its recovery time is relatively slow, it is ideal for low-frequency rectification applications, such as power supplies and protection circuits. This diode is widely used in electronic devices where high speed or minimal voltage drop is not required.

3.6 2N2222 Transistor

The 2N2222 is an NPN silicon transistor commonly used in amplification and switching applications. It is one of the most popular transistors due to its low cost, ease of use, and versatility. The 2N2222 has a maximum collector current of 800 mA and a maximum collector-emitter voltage of 40 V. Its current gain (hFE) typically ranges from 100 to 300, making it suitable for low-frequency circuits, signal amplification, and electronic device control. It is widely used in switching circuits, signal modulation, and as a switch to control devices such as relays, motors, and lights.

3.7 TRIAC BT138

The TRIAC BT138 is a solid-state electronic component used for controlling AC power. It is a triode for alternating current (TRIAC) with a maximum RMS voltage rating of 600 V and a maximum current rating of 4 A. The BT138 is commonly used in light dimmers, motor speed controllers, and other AC power regulation applications due to its ability to switch on and off during both positive and negative cycles of an AC signal. It has a high level of reliability, fast switching time, and low conduction losses, making it ideal for use in household and industrial electronic devices where efficient power control is needed.

3.8 MOC 3021 Optocoupler

The MOC 3021 is an optocoupler, or optoisolator, designed to provide electrical isolation between different sections of a circuit while allowing signal transmission via light. It is specifically used for isolating control signals from high-power circuits, protecting sensitive components from voltage spikes and noise. The MOC 3021 has a phototransistor output and operates with a low input current, typically used to drive TRIACs in AC power control applications. It is commonly employed in light dimmers, motor speed control, and solid-state relays. With a maximum collector-emitter voltage of 400 V and current rating of 50 mA, the MOC 3021 provides a safe and efficient way to interface low-voltage control circuits with higher voltage systems.

3.9 4N25 Optocoupler

The 4N25 is an optocoupler (or optoisolator) used to electrically isolate different sections of a circuit while transmitting signals via light. It consists of an LED and a phototransistor, providing isolation between input and output circuits. The 4N25 is commonly used in applications where it is necessary to protect sensitive components from high-voltage spikes or electrical noise. It can be used in control circuits, signal isolation, and switching applications, especially in envi-

ronments with high voltage, such as power supplies and microcontroller interfacing. With a maximum collector-emitter voltage of 35 V and current transfer ratio (CTR) typically ranging from 20% to 300%.

3.10 JQC-3FF-S-Z Relay

The JQC-3FF-S-Z is a small, general-purpose electromagnetic relay used for switching low to moderate power loads. It has a 5V DC coil voltage and is commonly used in applications where switching between different circuits or controlling high-power devices is needed. The relay features a single pole double throw (SPDT) configuration, meaning it has one input and two output terminals, allowing for both normally open (NO) and normally closed (NC) contacts. With a maximum switching current of 10A at 250V AC, the JQC-3FF-S-Z is suitable for controlling devices like motors, lights, and other appliances.

4 Operation of the Arduino UNO board

The Arduino Uno played a crucial role in collecting and processing data from the sensors used in the project. The sensors connected to the Arduino were responsible for measuring key environmental variables such as ambient temperature and humidity. For this purpose, sensors like the DHT11 for humidity and the LM35 for temperature were used. These sensors were connected to the digital and analog pins of the Arduino for reading their output values.

The process began with the periodic reading of the sensor values, using analog or digital reading functions depending on the type of sensor. For example, the LM35 temperature sensor provided an analog value proportional to the ambient temperature, which was then converted to degrees Celsius by the Arduino code. Similarly, the DHT11 humidity sensor delivered digital data that was processed by the microcontroller.

Once the values from all the sensors were obtained, the Arduino was responsible for sending this data to the Raspberry Pi for further processing and visualization. This was achieved via I2C communication, an efficient and low-cost protocol for data transmission between devices. The Arduino acted as the master in the I2C communication, sending the sensor values to the Raspberry Pi, which functioned as the slave. Through this link, temperature, humidity, and other measurements were continuously and in real-time transmitted to the Raspberry Pi.

This data flow enabled the Raspberry Pi to make decisions based on the Arduino's readings, controlling elements such as fans, water pumps, and lighting systems in the greenhouse. Additionally, the I2C communication allowed for minimizing the number of wires and simplifying the system design, as multiple devices could share the same communication bus.

5 Operation of the Raspberry Pi board

The Raspberry Pi was set up to function as the central server for the web-based user interface of the greenhouse control system. The HTML file for the web page, which allows the user to monitor and interact with the greenhouse system, was stored locally on the Raspberry Pi's file system. This file was then served over a local network using a lightweight web server, enabling the interface to be accessed from any device connected to the same network.

Additionally, the Raspberry Pi was configured to run Python scripts that acted as the communication bridge between the Arduino Uno and the web interface. These Python scripts were designed to handle data received from the Arduino, which was connected to various environmental sensors (such as temperature and humidity sensors). The data was transmitted from the Arduino to the Raspberry Pi using the I2C communication protocol, allowing real-time monitoring of the greenhouse conditions.

The Raspberry Pi received the sensor data sent by the Arduino and processed it as needed. This data was then displayed on the web interface, which was accessible to users through any browser on the local network. In this way, the Raspberry Pi not only served the web page but also acted as the intermediary that processed and visualized the sensor readings, facilitating the control and automation of the greenhouse environment.

6 Health and Safety Warnings

- **Heat Exposure:** Components like the *incandescent light bulb* used for lighting in the greenhouse and some of the power supply units can become hot during operation. To avoid the risk of *burns*, do not touch these components while they are in operation, and allow them to cool down before handling them.
- **Water Hazards:** Since the system includes a water pump for irrigation, it is important to ensure that no water comes into contact with electrical components to prevent the risk of *electrical shock*. Use waterproof enclosures for sensitive electronics, and regularly check that wiring is properly insulated.
- **Overheating and Fire Risk:** Improper wiring or a malfunctioning power supply can cause *overheating*, which may lead to system failures or even fires. To mitigate this risk, use circuit protection elements like fuses or circuit breakers, and ensure that the system is properly ventilated. Regularly check the temperature of components during operation.
- **Handling Sensitive Electronics:** The Raspberry Pi, Arduino, and sensors are sensitive electronic devices. Always handle them carefully to prevent physical damage. Use an *anti-static wrist strap* when working with these components to protect against electrostatic discharge (ESD), which can damage the electronics.
- **Ventilation:** The greenhouse environment must be well-ventilated to ensure the proper functioning of temperature sensors and to prevent overheating of the system components. Avoid blocking air vents or placing the system in poorly ventilated spaces to reduce the risk of thermal damage.
- **General Safety:** Keep the system away from flammable materials and ensure that it is installed in a safe, dry, and secure environment. Regularly inspect wiring and components for signs of wear or damage and replace them if necessary to maintain safe operation.

By following these guidelines and taking proper precautions, users can safely operate the greenhouse control system while minimizing health and safety risks.

7 Precautions for Handling Sensitive Electronic Components

In the greenhouse control system project, various sensitive electronic components were used, such as the Arduino UNO, Raspberry Pi, and various sensors, all of which require special care to ensure optimal performance and longevity. Proper handling and maintenance of these components are essential to avoid damage and malfunctions.

- **Avoiding Grime on Terminals:** Dust, grease, and other contaminants can accumulate on the terminals of components like the sensors, transistors, and relays. These contaminants can cause poor connections, leading to unreliable readings or malfunctioning of the system. To avoid this, always handle the components with clean hands or wear gloves. Ensure that connectors and terminals are free from dirt or oil, and clean them periodically with an appropriate electronics-safe cleaning solution or air blower.
- **Temperature Limits:** Many components, including the temperature sensors (such as the LM35) and Arduino UNO, have specific operating temperature ranges. Exceeding these limits may result in inaccurate readings or permanent damage to the components. For example, the LM35 temperature sensor has a typical operating range of -55°C to +150°C, while the Raspberry Pi has a maximum operating temperature of 85°C. Always ensure that the system operates within the recommended temperature limits and that proper ventilation is provided to avoid overheating.
- **Proper Storage:** When not in use, sensitive components such as the Raspberry Pi and Arduino should be stored in a dry, cool place, away from extreme temperatures or moisture. Avoid placing these components in environments with high humidity, as moisture can cause corrosion and short circuits.
- **Power Surge Protection:** Power surges and fluctuations can damage sensitive components like the Raspberry Pi and Arduino. It is recommended to use a surge protector or uninterruptible power supply (UPS) to protect the system from power spikes and to ensure stable voltage to the components.

- **Avoiding Physical Damage:** Components such as the optocouplers, relays, and sensors are sensitive to mechanical stress. Avoid subjecting these components to physical impact, bending, or excessive force. Store them in anti-static bags or compartments when not in use to prevent accidental damage.
- **Regular Inspections:** Regularly check the components and the overall system for any signs of wear, corrosion, or damage. Inspect connectors, wires, and solder joints to ensure that all connections are intact and properly made. Preventing small issues from escalating can prolong the lifespan of the system and avoid failure during operation.

8 Arduino Configuration

The Arduino UNO was used as the central controller for the greenhouse system. Below is the detailed configuration of the microcontroller, including the pins used, peripherals, and setup required for the system.

8.1 Hardware Configuration:

- **Microcontroller Used:** Arduino UNO
- **Digital Pins:**
  - **Pin 4:** Connected to the **DHT11** sensor (for humidity and temperature measurements). This pin is defined as the input pin for the sensor in the code.
  - **Pin 9:** Connected to the **fan** via PWM. This pin controls the fan speed based on the system's needs.
  - **Pin 7:** Connected to the **water pump**. This pin controls the activation of the water pump used for irrigation.

8.2 Peripheral Configuration:

- **DHT11 Sensor:** The **DHT11** sensor is used to measure temperature and humidity. It is connected to digital pin **4** of the Arduino. The sensor is initialized with the `DHT.h` library to manage the communication and data retrieval.
- **I2C Communication (Wire Library):** The Arduino UNO communicates with the Raspberry Pi via the **I2C** protocol. The Arduino is configured as an I2C slave with address **8** (which can be adjusted in the code as needed). The `Wire.h` library is used to handle the communication, and the `Wire.onReceive()` function is called to handle incoming data and commands.

8.3 Libraries Used:

- **DHT.h:** This library is used to interface with the **DHT11** sensor. It allows for reading the temperature and humidity data.
- **Wire.h:** This library facilitates I2C communication, allowing the Arduino to receive commands and transmit data to the Raspberry Pi.

9 Raspberry Pi Configuration

The Raspberry Pi serves as the central hub for managing the greenhouse control system. It is responsible for hosting the web interface, handling communication with the Arduino, and controlling the system through the sensors and devices (fan, water pump). The Raspberry Pi was configured with Node.js to serve the web page, handle I2C communication, and interact with the Arduino UNO for real-time monitoring and control.

9.1 Hardware Configuration

The Raspberry Pi used in this project is a single-board computer running Raspberry Pi OS. It was connected to the local network, allowing remote access to the web interface from any device on the same network.

The Raspberry Pi is connected to the Arduino UNO via the I2C communication protocol. This connection enables data exchange between the two devices, allowing the Raspberry Pi to send commands to the Arduino to control the fan and water pump, and to receive sensor data (temperature, humidity) from the Arduino.

9.2 Software Configuration

The software configuration of the Raspberry Pi includes the installation and setup of several key com-

ponents:

- **Raspberry Pi OS:** The operating system used for this project, which provides the necessary environment to run the web server and manage I2C communication.
- **Node.js:** The web server is built using Node.js, which is installed on the Raspberry Pi to serve the HTML page and handle HTTP requests. Node.js was chosen because of its lightweight nature and efficiency in handling asynchronous operations, which is essential for real-time data processing.
- **Express.js:** The Express.js framework is used to simplify the setup of the web server. It handles routing and manages the HTTP requests sent by the web interface. Express is also responsible for sending data to the Arduino and receiving responses.
- **Web Server:** The Raspberry Pi hosts a local web server that runs on port 3000 (as configured in the Node.js server). This server serves the HTML, CSS, and JavaScript files for the web interface. The web page allows users to monitor real-time data from sensors (e.g., temperature and humidity), as well as control devices like the fan and water pump.
- **I2C Communication:** The Raspberry Pi is set up to communicate with the Arduino via I2C. The `smbus` library is used on the Raspberry Pi to manage I2C communication. This allows the Raspberry Pi to send control commands to the Arduino (e.g., turning on the fan or the water pump) and to receive sensor data from the Arduino (e.g., temperature and humidity readings).
- **Python (Optional):** In some cases, Python is used to manage the I2C communication, especially if more complex data processing or additional control is needed.

9.3 Steps for Setup

- **Install Raspberry Pi OS:** The Raspberry Pi OS is installed and configured on the Raspberry Pi. The system is set up to connect to the local Wi-Fi network to allow access to the web interface from any device on the same network.
- **Install Node.js and Express:**
  - Install Node.js on the Raspberry Pi using the following commands:

```
sudo apt update
sudo apt install nodejs
sudo apt install npm
```

  - Install Express.js to create the web server:

```
npm install express
```

- **Set Up the Web Server:**
  - The HTML page and associated files (CSS and JavaScript) are stored in a directory on the Raspberry Pi.
  - The Node.js server is set to serve the files and handle HTTP requests from the client (web page).
  - The server listens on port 3000, and the web page can be accessed through `http://<raspberry-pi-ip>:3000`.
- **Enable I2C Communication:**
  - The I2C interface is enabled on the Raspberry Pi via the `raspi-config` tool:

```
sudo raspi-config
```

  - Select `Interfacing Options` and enable I2C.
  - Install the required I2C libraries:

```
sudo apt-get install
i2c-tools python-smbus
```

- **Communication with the Arduino:**
  - The I2C address of the Arduino is configured as 8, allowing the Raspberry Pi to send and receive data through the Wire library.
  - The Raspberry Pi sends control commands (e.g., turning the fan on or off) to the Arduino, and the Arduino responds with the sensor data (e.g., temperature and humidity readings).
- **Testing and Debugging:**
  - The system was tested by controlling the fan and water pump through the web interface and confirming that the data from the sensors (temperature and humidity) was updated correctly in real-time.

– The Serial Monitor on the Raspberry Pi was used to view debug output and ensure proper communication with the Arduino.

9.4 Networking Configuration

The Raspberry Pi is connected to the local network via Wi-Fi or Ethernet, and the web server is configured to be accessible from any device connected to the same network. By accessing `http://<raspberry-pi-ip>:3000`, users can interact with the web page, monitor sensor data, and control the greenhouse environment remotely.

10 Development of the Web Page Software Module

The web page for the greenhouse control system was developed to provide a graphical user interface (GUI) that allows users to remotely control and monitor the greenhouse environment. This software module consists of three main components: HTML for structure, CSS for styling, and JavaScript for functionality. Below is a description of each component and its role in the system.

The HTML file provides the basic structure of the web page, defining visual elements such as buttons, text fields, labels, and sections where sensor readings (such as temperature and humidity) are displayed. The HTML file also sets up the layout and hierarchy of the page's elements. One of the main features of the HTML file is the control form for devices, which includes buttons that allow the user to turn the fan and water pump on or off. These buttons send corresponding commands to the Raspberry Pi, which then communicates with the Arduino to execute the actions. Additionally, the HTML file facilitates the dynamic display of real-time sensor data, which is updated using JavaScript to reflect the latest readings.

The CSS file is responsible for defining the visual appearance of the page. It controls aspects such as the layout, colors, fonts, and positioning of elements on the screen. The CSS ensures that the interface is clean, visually appealing, and easy to use. The page is designed to be responsive, ensuring that it displays well on both desktop computers and mobile devices. The CSS also styles the buttons and controls, making them easily identifiable and interactive, with colors indicating their active or inactive states. The overall design is clean and organized, allowing for easy navigation and use of the interface.

JavaScript provides the interactivity of the page, allowing the web interface to respond to user input dynamically. It is responsible for sending commands to the Node.js server, receiving real-time updates from sensors, and dynamically updating the user interface. When a user presses a button to turn on the fan, for example, JavaScript sends a command to the server, which processes it and communicates with the Arduino to execute the action. The server then responds with the updated status, and JavaScript updates the UI accordingly.

The primary function of JavaScript is to send commands to the Raspberry Pi via HTTP requests. These commands can turn devices on or off, such as the fan or water pump. JavaScript also handles the updating of sensor data on the web page, ensuring that the user always sees the most recent information. The communication between the web page and the Raspberry Pi is managed by the server running on the Raspberry Pi, which communicates with the Arduino to control the hardware.

11 Conclusions

The greenhouse automation project successfully integrated a variety of electronic components and technologies to create a functional and efficient system for monitoring and controlling environmental variables such as temperature, humidity, and water levels. The system utilized an Arduino UNO as the primary controller for reading sensor data, while the Raspberry Pi managed the communication and provided an interface for remote control through a web page. The integration of the DHT11 sensor, water pump, fan, and other components with the Arduino UNO demonstrated the effectiveness of using a microcontroller for reading environmental data and controlling various peripherals. The system successfully managed to read temperature and humidity values, activate or deactivate the fan, and control the water pump based on the system's needs.

The use of I2C communication between the Arduino and Raspberry Pi enabled seamless data transfer, ensuring that the Raspberry Pi could efficiently receive sensor data and send control commands to the Arduino, providing the necessary functionality for the

system. The development of a web-based user interface allowed easy monitoring and control of the greenhouse system. This interface empowered users to make adjustments to the environment from anywhere with an internet connection, making the system more flexible and accessible. While the current system provides a solid foundation, there is room for future improvements and expansions. Additional sensors (e.g., soil moisture, light levels) could be integrated into the system for more comprehensive environmental control. Furthermore, the system could be optimized for energy efficiency and reliability. In conclusion, this project highlights the practical applications of embedded systems in real-world environments, demonstrating how automation can be applied to agriculture and other sectors requiring environmental monitoring.

12 Bibliography

- Arduino. (2024). *What is Arduino?* Arduino Documentation. Retrieved November 7, 2024, from `https://docs.arduino.cc/learn/starting-guide/whats-arduino/`
- Balci, M. E., & Hocaoglu, M. H. (2005, November). Effects of source voltage harmonic distortion on power factor compensation in triac controlled AC chopper circuits. In *2005 International Conference on Power Electronics and Drives Systems* (Vol. 2, pp. 1199-1204). IEEE.
- Guijarro-Rodríguez, A. A., Cevallos-Torres, L., PRECIADO-MAILA, D. K., & Zambrano-Manzur, B. N. (2018). *Sistema de riego automatizado con Arduino. Sistema,* 39(37), 27.
- Karthikeyan, P., Karthik, M., Deepik...
- Ota, S. B., & Nanda, K. K. (1994). *The temperature dependence of the forward characteristics of 1n4007 silicon diode.* Review of scientific instruments, 65(10), 3289-3290.
- Raj, B. P., Aradhyula, E., Swamy, P. B., & Pindoo, I. A. (2021, April). *TRIAC Based Home Automation System with User Friendly Interface.* In 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) (pp. 391-395). IEEE.
- Raspberry Pi Foundation. (2024). *Raspberry Pi documentation: Introduction. Raspberry Pi.* Retrieved November 26, 2024, from `https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#introduction`
- Slotnick, B. (2009). *A simple 2-transistor touch or lick detector circuit.* Journal of the experimental analysis of behavior, 91(2), 253.
- Valdez, J., & Becker, J. (2015). *Understanding the I2C bus.* Texas Instruments.

References

[1] Youtube video `https://youtu.be/qhnyMYJImB8`

[2] Github Repository `https://github.com/AmandaBalderas/Greenhouse-Project`