**Survey on Raspberry Pi and Arduino – Comparison and Applications**

**Effat University**  
**Spring 2025**

**Students names:** Amro Yousef, Layan Turkistani, Tariq Tadmori

**Code Name:** Analog control system  
**course code:** ECE 342L  
**Instructor:** Dr. Nema Salem

**Table of Contents**

1. **Introduction……………………………………………………………………..……………..3**
2. **Objective……………………………………………………………………………………….3**
3. **Abstract…………………………………………………………………………………….…..3**
4. **Comparison of Raspberry Pi and Arduino……………………………………….…….3,4,5,6**
5. **Applications of Raspberry Pi and Arduino………………………………………………..6,7**
6. **Implementation Plan: Smart Irrigation System using Raspberry Pi…………...…..7,8,9,10**
7. **Conclusion…………………………………………………………………………………....11**
8. **References…………………………………………………………………………….……11,12**
9. **Introduction**

Embedded systems and microcontrollers play a crucial role in modern automation and control applications. Among the various platforms available, Raspberry Pi and Arduino are two of the most widely used. While Arduino is known for its simplicity and efficiency in real-time applications, Raspberry Pi offers greater computational power and connectivity options, making it suitable for more complex tasks. This report provides a comparative analysis of these platforms, discussing their features, advantages, and limitations in control engineering. Additionally, a practical implementation of a Smart Irrigation System using Raspberry Pi is presented to demonstrate its real-world applicability.

1. **Objective**

The objective of this study is to conduct a comprehensive survey and comparison between Raspberry Pi and Arduino, analysing their hardware and software capabilities, control engineering applications, and suitability for embedded systems. Furthermore, an implementation plan for a Smart Irrigation System using Raspberry Pi will be detailed to demonstrate its practical application.

1. **Abstract**

Raspberry Pi and Arduino are widely used platforms in embedded systems and automation. This report compares their specifications, programming environments, and best use cases, highlighting their roles in home automation, industrial automation, and robotics. The proposed Smart Irrigation System leverages Raspberry Pi’s computational power and connectivity to optimize water usage, showcasing its superiority in IoT-based control applications. The findings suggest that while Arduino is ideal for real-time control tasks, Raspberry Pi is better suited for more complex applications requiring multitasking and network integration.

1. **Comparison of Raspberry Pi and Arduino**

Raspberry Pi and Arduino are two of the most widely used platforms for embedded systems and control applications, but they serve different purposes due to their distinct architectures and capabilities.

**Strengths and Weaknesses of Raspberry Pi**

1. **Strengths:**

* **High computational power**: Equipped with ARM-based processors, Raspberry Pi can handle multitasking, advanced computing, and AI applications.
* **Supports multiple programming languages**: Python, C++, Java, and others allow flexibility in software development.
* **Networking capabilities**: Built-in Wi-Fi, Bluetooth, and Ethernet provide seamless communication for IoT applications.
* **Multimedia processing**: Supports HDMI output, video streaming, and graphic processing.
* **Expandable storage**: Uses SD cards and external drives for scalable storage needs.

1. **Weaknesses:**

* **Higher power consumption**: Requires a constant 5V power supply with at least 2-3A, making battery-powered operation less efficient.
* **Lacks built-in analog input support**: Requires an external ADC (Analog-to-Digital Converter) module for reading analog sensors.
* **Overhead of an operating system**: Running a full Linux-based OS introduces boot-up time and system lag compared to real-time microcontrollers.
* **More complex programming**: Requires knowledge of Linux and shell scripting for advanced configurations.

**Strengths and Weaknesses of Arduino**

1. **Strengths:**

* **Real-time processing**: Executes commands instantly without the delay of an operating system.
* **Low power consumption**: Operates efficiently at 3.3V or 5V, making it suitable for battery-operated applications.
* **Built-in analog input**: Directly supports analog sensors without the need for an external ADC.
* **Easy to program**: Uses a simple C/C++ based environment (Arduino IDE), making it accessible for beginners.
* **Robust for embedded systems**: Well-suited for standalone automation, sensor monitoring, and motor control.

1. **Weaknesses:**

* **Limited processing power**: With a clock speed ranging from 16 MHz to 84 MHz, it cannot handle computationally intensive tasks.
* **No built-in networking**: Lacks Wi-Fi, Bluetooth, or Ethernet without external modules.
* **Single-task execution**: Cannot run multiple processes simultaneously, limiting multitasking capabilities.
* **Limited memory and storage**: Flash memory is typically between 32KB and 256KB, restricting large applications.

**Processor and Computing Power:**

Raspberry Pi is equipped with an ARM-based processor capable of running a full Linux operating system, making it a mini-computer suitable for multitasking, data analysis, and AI applications. In contrast, Arduino relies on microcontrollers, which are optimized for executing real-time, single-threaded applications with minimal processing overhead.

**Memory and Storage:**

Raspberry Pi models come with 512MB to 8GB RAM, supporting applications requiring large data storage and real-time processing. It uses SD cards or external USB drives for storage, allowing flexibility in software deployment. Arduino, on the other hand, has a much smaller memory capacity, typically between 2KB and 8KB of SRAM, and uses flash memory for storing programs. This makes it less suited for tasks requiring data-intensive operations.

**Operating System and Programming:**

Raspberry Pi supports Linux-based operating systems such as Raspberry Pi OS and Ubuntu, allowing the use of high-level programming languages like Python, C++, Java, and JavaScript. It is capable of running complex applications with a graphical user interface. Arduino, however, lacks an operating system and executes single-threaded C or C++ programs directly on the microcontroller, making it ideal for low-level hardware control but limited in multitasking.

**Connectivity and Peripherals:**

One of Raspberry Pi’s biggest advantages is its built-in connectivity options, including Wi-Fi, Bluetooth, Ethernet, USB, and HDMI. These features enable it to function as an IoT gateway or web server. Arduino has limited connectivity, requiring additional modules (e.g., Wi-Fi or Bluetooth shields) to achieve network capabilities. However, Arduino excels in analog and real-time interfacing, offering direct analog input support, whereas Raspberry Pi requires an external ADC module to process analog sensor data.

**Power Consumption and Energy Efficiency:**

Raspberry Pi consumes higher power (5V, 2-3A) due to its processing capabilities and multiple onboard peripherals. This makes it more power-hungry and less suitable for battery-operated devices. Arduino, in contrast, operates at very low power (3.3V or 5V, typically consuming milliamps), making it an excellent choice for energy-efficient applications such as remote sensor networks and low-power IoT devices.

**Security Features:**

With its Linux-based architecture, Raspberry Pi supports encryption, firewall configuration, and secure remote access features. These security measures make it suitable for IoT applications requiring cloud-based data management and secure communications. Arduino, being a microcontroller-based platform, has limited security capabilities and relies on external encryption techniques for data protection.

**Use Cases and Application Suitability:**

* Raspberry Pi is well-suited for IoT applications, AI-based projects, multimedia processing, and automation due to its high computing power and built-in network support.
* Arduino is preferred for real-time sensor interfacing, robotics, automation control, and energy-efficient projects, where minimal processing power is required.

**Technical Comparison Table:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Raspberry Pi** | **Arduino** |
| **Processor** | ARM-based (varies by model, e.g., Cortex-A72, Quad-Core) | Microcontroller (AVR, ARM Cortex-M series) |
| **Clock Speed** | 700 MHz - 1.8 GHz | 16 - 84 MHz |
| **Memory** | 512MB - 8GB RAM | 2KB - 8KB SRAM (varies by model) |
| **Storage** | SD card (expandable, up to 1TB), supports external USB drives | Flash memory (limited, e.g., 32KB - 256KB) |
| **Operating System** | Linux-based OS (Raspberry Pi OS, Ubuntu, etc.) | No OS, runs single programs directly |
| **Programming Languages** | Python, C++, Java, Scratch, Shell scripting, JavaScript | C, C++ (Arduino IDE, Platform IO) |
| **I/O Pins** | 26-40 GPIO pins with PWM support, supports SPI, I2C, UART | 6-54 Digital I/O, 6-16 Analog I/O (varies by model) |
| **Analog Input Support** | Requires external ADC module | Built-in ADC for analog sensor readings |
| **Connectivity** | Wi-Fi, Bluetooth, Ethernet, USB, HDMI, GPIO | USB, UART, I2C, SPI (Wi-Fi/Bluetooth via shields) |
| **Power Consumption** | Higher (5V, 2-3A, depends on model) | Low power (operates on 5V or 3.3V) |
| **Multitasking** | Supports multitasking, runs multiple processes concurrently | Runs one program at a time |
| **Peripheral Support** | Supports cameras, displays, external storage, AI accelerators | Limited to basic sensors, shields, and actuators |
| **Security Features** | Supports encryption, firewall, secure boot options | Minimal security features due to hardware limitations |
| **Best Use Case** | AI, multimedia, automation, IoT, robotics, data logging, edge computing | Real-time control, sensors, robotics, automatio |

1. **Applications of Raspberry Pi and Arduino**

**Home Automation**

Home automation has become a popular field for embedded systems, with Raspberry Pi and Arduino playing crucial roles in implementing smart home solutions.

* **Raspberry Pi**: Acts as a smart home hub, integrating IoT devices through Wi-Fi, Bluetooth, or Ethernet. It can control lighting, HVAC systems, smart securitycameras, and voice assistants. It supports platforms like Home Assistant or allowing users to automate multiple devices in their homes. Additionally, with Python scripts, users can create custom automation sequences.
* **Arduino**: Works best for basic automation tasks, such as controlling individual components like motion-activated LED lights, automated window blinds, and temperature-based fan control. It is particularly useful for real-time, low-power tasks where networking is not necessary.

**Industrial Automation**

Industrial automation has significantly benefited from microcontroller and microprocessor-based systems, improving efficiency, safety, and monitoring.

* **Raspberry Pi**: Used for process automation, real-time data monitoring, and predictive maintenance in factories. It supports SCADA (Supervisory Control and Data Acquisition) systems, allowing engineers to monitor and control industrial operations remotely. It can also integrate with cloud services for data logging and AI-driven process optimization.
* **Arduino**: Commonly used for machine control, sensor interfacing, and relay actuation in industrial systems. It is embedded into assembly lines, motor controllers, conveyor belts, and factory robotics. While it does not support high-level data processing, it is a reliable, real-time controller for mechanical processes.

**Robotics**

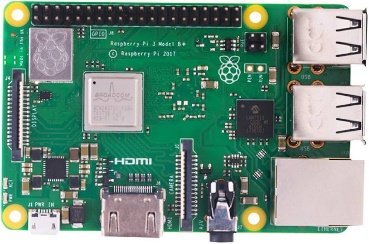
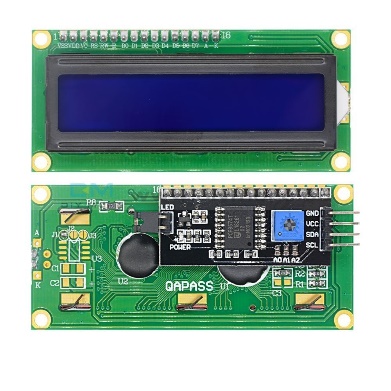
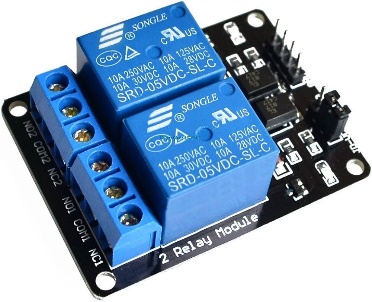
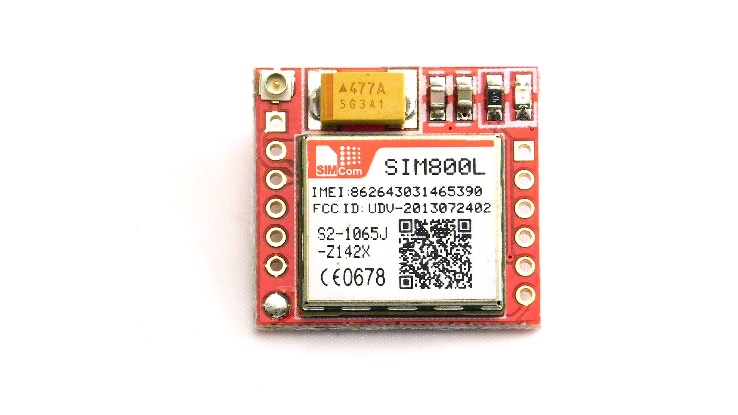
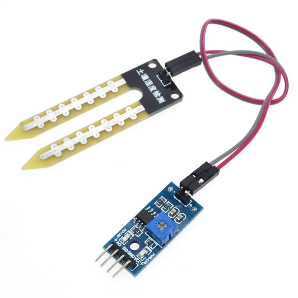
Robotics is an interdisciplinary field that heavily relies on embedded systems.

* **Raspberry Pi**: Used for AI-driven robotic navigation, object detection, and automation. It supports vision-based tasks using OpenCV and machine learning models, making it useful for applications such as autonomous vehicles, drones, and robotic arms. The Robot Operating System (ROS) can be installed on Raspberry Pi, allowing developers to build complex robotic projects.
* **Arduino**: Works well for low-level motor control and basic autonomous robot projects, such as line-following robots, robotic arms, and gesture-controlled vehicles. It is widely used in robotic competitions and educational projects.

1. **Implementation Plan: Smart Irrigation System using Raspberry Pi**

To develop a Smart Irrigation System that automatically waters plants based on soil moisture levels while optimizing water usage and enabling remote monitoring.

**Hardware Components:**

1. **Raspberry Pi 3 Model B+ UK** 
   * Serves as the main processing unit for the system.
   * Reads sensor data, processes it, and controls components like the water pump and GSM module.
   * Supports remote monitoring and IoT integration through Wi-Fi and Bluetooth connectivity.
2. **LCD 2×16 with I2C**
   * Displays real-time data, such as soil moisture level, temperature, and system status.
   * Uses I2C communication, reducing the number of GPIO pins required for connection.
   * Helps with on-site monitoring without needing an external display.
3. **Relay Module (2-Channels)** 
   * Acts as an electrical switch to control the water pump based on sensor readings.
   * Provides electrical isolation, ensuring safe operation when switching high-power loads.
   * Supports AC and DC devices, making it versatile for various applications.
4. **GSM Module (SIM800L)**
   * Enables remote communication via SMS or mobile networks.
   * Sends alerts or system updates (e.g., low moisture levels) to a mobile device.
   * Can be used for remote control of the irrigation system via SMS commands.
5. **Large 10mm LED (Blue)** 
   * Serves as an indicator for different system statuses.
   * Can be programmed to light up when the pump is active or to indicate system errors.
   * Helps with quick visual diagnostics of the system’s operation.
6. **Fan 5V for Raspberry Pi** 
   * Keeps the Raspberry Pi cool, preventing overheating during extended operation.
   * Essential for maintaining performance, especially in outdoor or warm environments.
   * Helps prolong the lifespan of the Raspberry Pi under high computational loads.
7. **Soil Moisture Water Sensor**
   * Measures soil moisture levels in real time.
   * Sends data to the Raspberry Pi, which decides when to activate the water pump.
   * Works by detecting the soil's conductivity, providing accurate irrigation control.
8. **Light Sensor (LDR Module)** 
   * Detects ambient light levels, which can be used for smart irrigation optimization.
   * Helps determine if watering is needed based on sunlight conditions.
   * Can also be used for additional automation features, such as activating lighting.
9. **LM35 DZ Temperature Sensor**
   * Measures the environmental temperature around the irrigation area.
   * Provides data to optimize watering schedules based on temperature variations.
   * Offers high accuracy and low power consumption, making it ideal for embedded systems.
10. **Power Supply (5V, 2A or higher, depending on Raspberry Pi model)**

* Provides a stable power source for the Raspberry Pi and other connected components.
* Can be a power adapter, battery pack, or solar-powered system for remote applications.
* Ensures uninterrupted operation, especially for IoT-based monitoring.

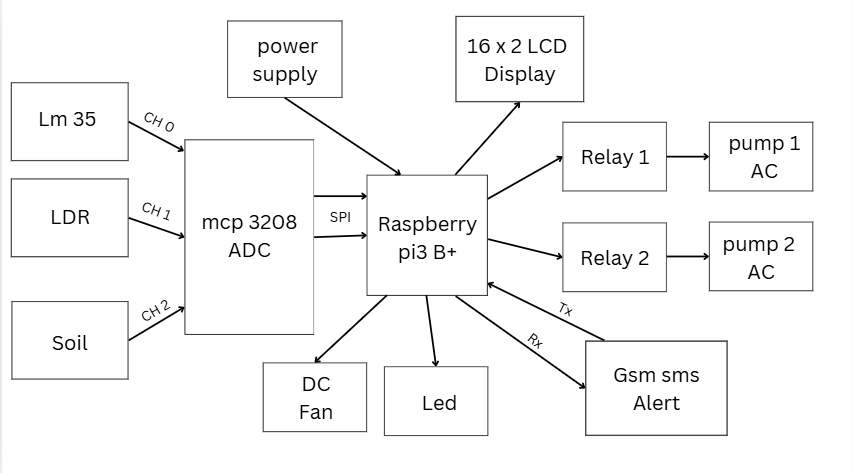
**Software Tools and Programming Languages:**

* **Programming Language:** Python
* **Libraries:** GPIO, Adafruit libraries for sensors
* **OS:** Raspberry Pi OS

**System Architecture:**

1. The soil moisture sensor collects data and sends it to Raspberry Pi.
2. Based on predefined thresholds, Raspberry Pi decides whether to turn the water pump on/off.
3. The relay module controls the water pump.
4. The LCD displays system status (e.g., moisture level, pump status).

**Block diagram**



**Why Raspberry Pi for this Project?**

Raspberry Pi is chosen over Arduino for the Smart Irrigation System due to its ability to process real-time data, integrate with network-based monitoring, and support multitasking. Unlike Arduino, which is ideal for simple sensor-actuator-based control, Raspberry Pi offers:

* **Advanced Processing Power:** Enables real-time data logging, analytics, and control.
* **Built-in Connectivity:** Supports Wi-Fi, Ethernet, and Bluetooth, allowing remote monitoring.
* **Support for Multiple Sensors and Actuators:** Can handle multiple input/output operations efficiently.
* **Flexibility in Software Implementation:** Runs Python scripts with complex logic, unlike Arduino’s C-based programming environment.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Raspberry Pi** | **Arduino** |
| **Data Processing** | Handles real-time data analysis | Limited processing power |
| **Connectivity** | Supports Wi-Fi, Ethernet, Bluetooth | No built-in connectivity |
| **Multitasking** | Can run multiple processes | Runs single tasks at a time |
| **Remote Monitoring** | Supports IoT & cloud integration | Requires additional modules for connectivity |
| **User Interface** | Can display data on screens | Limited display options |

**Expected Outcomes:**

* **Efficient Water Usage:** The system optimizes irrigation by watering only when necessary.
* **Automated and Smart Control:** Reduces human intervention and enhances agricultural efficiency.
* **Data Logging and Remote Monitoring:** With network integration, users can track system performance remotely.
* **Sustainability and Cost Reduction:** Reduces water waste and promotes sustainable irrigation practices.
* **Scalability:** The system can be expanded to include multiple sensors and zones.

**Challenges:**

* **Component Durability and Environmental Factors:** Ensuring sensors and actuators remain functional in outdoor conditions.
* **Power Supply Stability:** Consistent power is needed for continuous operation.
* **Network Dependency:** Remote monitoring requires a stable internet connection.
* **Sensor Accuracy and Calibration:** Ensuring precise moisture level readings to avoid under or overwatering.
* **Integration with Cloud Services:** Managing and processing data efficiently for future improvements.

1. **Conclusion**

Both Raspberry Pi and Arduino are valuable platforms in embedded systems. Arduino excels in real-time control applications, while Raspberry Pi is better suited for complex and IoT-integrated projects. The Smart Irrigation System demonstrates the capability of Raspberry Pi in automation, contributing to sustainable water management. This study highlights the importance of selecting the appropriate platform based on the application’s computational and connectivity needs.

1. **References**
2. Upton, E., & Halfacree, G. (2021). *Raspberry Pi User Guide (4th ed.).* Wiley.
3. Banzi, M., & Shiloh, M. (2015). *Getting Started with Arduino (3rd ed.).* Maker Media.
4. Monk, S. (2022). *Programming the Raspberry Pi: Getting Started with Python (3rd ed.).* McGraw-Hill Education.
5. Margolis, M. (2020). *Arduino Cookbook (2nd ed.).* O'Reilly Media.
6. Ali, S., & Hassan, M. (2021). “Smart Irrigation System Using IoT and Raspberry Pi,” *International Journal of Smart Agriculture Technology*, 6(2), 55-68.
7. Raspberry Pi Foundation. (2023). *Raspberry Pi Documentation.* Retrieved from https://www.raspberrypi.org/documentation/
8. Arduino. (2023). *Arduino Reference and Tutorials.* Retrieved from https://www.arduino.cc/reference/en/
9. Sharma, P., & Singh, R. (2022). “A Comparative Study on Raspberry Pi and Arduino for IoT Applications,” *Journal of Embedded Systems and Robotics*, 10(1), 15-28.