Week 1 Project Report

Brahmaji Chukka

August 1, 2025

Abstract

This report outlines the work completed during the first week of the project. We have introduced the main project components, including a study of the ESP32 development board and the initial project plan.

1 Introduction

The objective of this project is to create a system that... *(Insert your project's overall goal here).* During this first week, the primary focus was on foundational research and understanding the key hardware component, the ESP32 development board.

2 Study on the ESP32 Development Board

The ESP-WROOM-32 is a development board based on the ESP32 microcontroller. It is a versatile and powerful board suitable for a wide range of IoT projects. The pin definitions for the development board were studied, which include various GPIO pins, ADC pins, and touch-enabled pins. The board also has dedicated pins for communication protocols such as I2C and SPI, as well as a built-in antenna for wireless communication.

2.1 Key Features

The ESP32 is a highly integrated and feature-rich Wi-Fi and Bluetooth-capable MCU. It includes:

- A dual-core processor.
- Integrated Wi-Fi and Bluetooth connectivity.
- A wide range of peripherals, including GPIO, ADC, and DAC.
- A low-power design, making it suitable for battery-powered applications.

3 Thonny Software Setup

Thonny is an integrated development environment (IDE) for Python that is particularly well-suited for beginners and for use with microcontrollers like the ESP32. This section outlines the steps to set up Thonny for programming the ESP32.

3.1 Installation

1. Download Thonny from the official website: `https://thonny.org/`.
2. Run the installer and follow the on-screen instructions to complete the installation.

3.2 Configuring Thonny for ESP32

1. Connect your ESP32 board to your computer via a USB cable.
2. Open Thonny.
3. Go to `Tools -> Options...` and select the `Interpreter` tab.
4. From the dropdown menu, select `MicroPython (ESP32)`.
5. Thonny should automatically detect the correct port. If not, you may need to select it manually from the `Port` dropdown list.
6. Click `OK` to save the settings.
7. A successful connection will show a MicroPython prompt in the Thonny shell.

4 Example Code Snippet

This section provides a simple MicroPython code snippet that can be run in Thonny to control an LED connected to an ESP32. This example demonstrates how to use the `machine` library to control a GPIO pin.

4.1 Code for Blinking an LED

This code will blink an LED connected to GPIO pin 2 of the ESP32 at a one-second interval.

```
from machine import Pin
import time

# Set up the LED pin as an output
led = Pin(2, Pin.OUT)

# Loop forever
while True:
    led.value(1) # Turn the LED on
    time.sleep(1) # Wait for 1 second
    led.value(0) # Turn the LED off
    time.sleep(1) # Wait for 1 second
```

Listing 1: MicroPython code to blink an LED

4.2 Expected Outcome

When this code is uploaded and run on the ESP32 using Thonny, the following will occur:

- The LED connected to GPIO pin 2 of the ESP32 will turn on for one second.
- It will then turn off for one second.
- This cycle of turning on and off will repeat indefinitely.

5 Project Workflow Flowchart

The initial project plan is represented in the flowchart below. This chart illustrates the general steps from data acquisition to the final output. This general flow is inspired by the system structure of an elevator health diagnosis system.

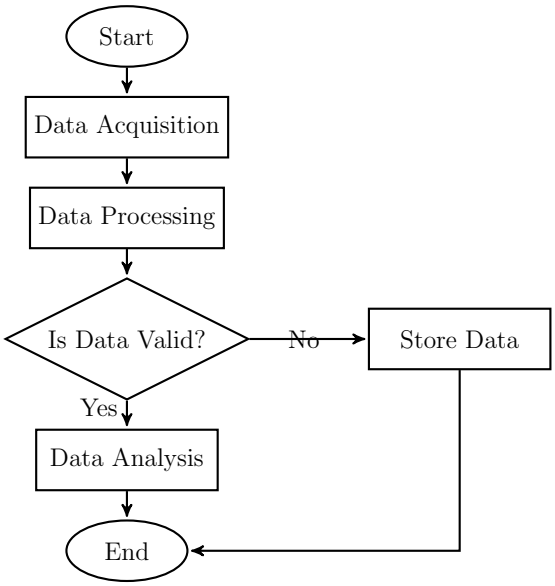

Figure 1: Project Workflow Flowchart

Week 2 Project Plan

Objective

The main objective for Week 2 is to begin the physical implementation of the project by integrating hardware components, writing and testing the core code, and verifying basic functionality.

Tasks for the Week

Task 1: Hardware Integration and Testing

- Action: Adding any integrated sensors for vibration Reading.
- Action: Connect the sensor to the ESP32 board, following the pinout definitions you studied in Week 1.

- Action: Test the hardware connection with a simple program to ensure the sensor is powered correctly and the wiring is sound.

Task 2: Develop and Test Initial Code

- Action: Using Thonny, write a MicroPython script to read data from the connected sensor. This will be an expansion of the simple LED blink program from Week 1.
- Action: Implement basic data processing, such as converting raw sensor readings into meaningful units (e.g., voltage to temperature, distance, etc.).
- Action: Test the code thoroughly to ensure it is correctly reading data from the sensor and displaying it in the Thonny shell.

Task 3: Implement Core Functionality

- Action: Start writing the code for a key feature of your project. This could be data logging, sending data to a cloud service (if applicable), or triggering an action based on sensor readings.
- Action: Conduct initial testing to see if the core functionality works as expected.

Deliverables for Week 2

- A working breadboard or circuit diagram showing the ESP32 connected to your primary sensor.
- A clean, well-commented Python script that reads data from the sensor and performs a basic action.
- A new section for your report detailing the hardware setup, the code, and the results of your initial tests.

6 Conclusion

The first week of the project has been successful in laying the groundwork for the development phase. The research on the ESP32 has provided a solid understanding of the hardware, and the initial project workflow has been established .