Assignment 2
Dew Point Generator Controller and Sensor Data Logger

James Cook University Cairns

Hunter Kruger-Ilingworth (14198489)
Quentin Bouet (14198423)
Thomas Mehes (14259613)

October 25, 2024

Contents

List of Figures

List of Tables

Intro

This report details the design and implementation of an embedded system functioning as both a scientific data logger and a smart interface with an analogue dew point generator. Utilizing the RP2040 microcontroller, SDI-12 environmental sensors, and a load cell, the system collects and processes data for climate modelling. The project aims to offer a reliable and efficient solution for monitoring and controlling environmental parameters, especially in researching tropical plant behavior under varying climate conditions.

The LI-610 Dew Point Generator is a precision instrument that produces a stable gas stream with a controlled dew point. It employs Peltier thermoelectric coolers to regulate water reservoir temperatures, ensuring the air stream is fully saturated with water vapor. This precise dew point control is vital in environmental research, preventing condensation in climate-controlled chambers and maintaining experimental conditions and data accuracy.

Accurate, continuous monitoring of environmental parameters is crucial for tropical plant research, but traditional methods are labor-intensive, error-prone, and physical presence in climate-controlled rooms can disrupt experiments. Commercial solutions like Campbell Scientific are often expensive and closed-source, limiting accessibility and customization for researchers. This project provides an open-source, cost-effective alternative for remote monitoring and control. By integrating various sensors and a smart interface for the dew point generator, the system enables the simulation of different climate conditions and monitoring of plant responses without entering the controlled environment, ensuring data integrity while enhancing flexibility and affordability.

System Design Overview

The system was designed around the RP2040 microcontroller, acting as the central processing unit for interfacing with a variety of sensors and control devices. The design incorporates the SF-5M sap flow sensor, LT-1T leaf temperature sensor, and MT-603 load cell to measure various environmental parameters. Each of these sensors use different communication protocols, requiring careful consideration of power requirements, signal integrity, and communication compatibility. A photo of the board is shown in fig. 2 while fig. 1 shows an overview of all the communication systems involved in the design.

Hardware Design

Load Cell and ADC Integration

The MT-603 load cell requires an ADC (Analogue-to-Digital Converter) for signal conversion, as its output is an analogue signal representing weight measurements. Therefore, one of the RP2040's ADC pins was utilised to convert the load cell's voltage output into a digital signal in which the microcontroller can process. An instrumentation amplifier was also added to amplify the small voltage changes from the load cell, improving the resolution and accuracy of the measurements. This ADC integration allowed for precise scaling and calibration of the load cell to achieve accurate data representation.

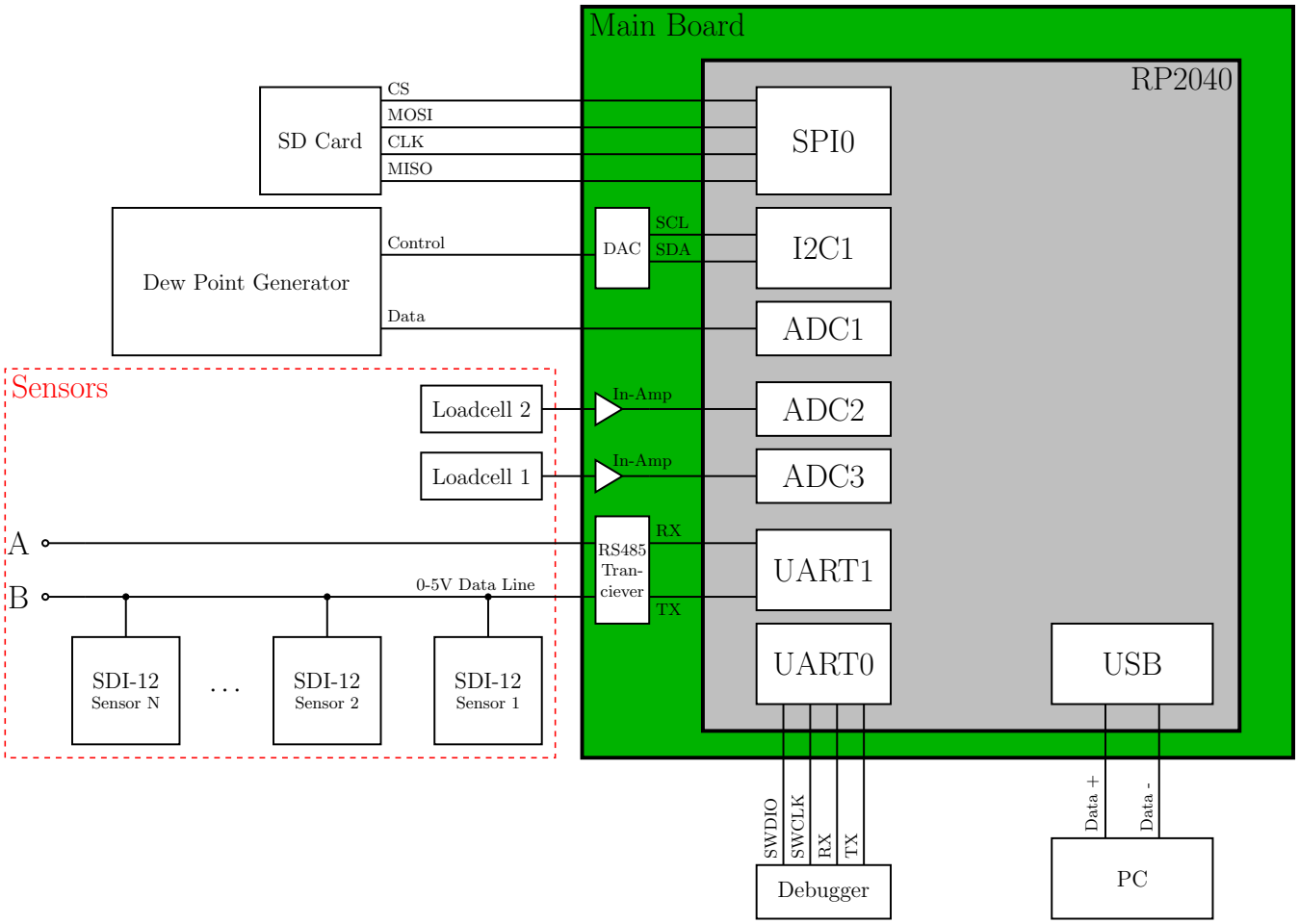

Fig. 1. Block diagram of the system

SDI-12 Sensors and Communication Interface

The SDI-12 protocol was implemented because it is the standard communication protocol for the environmental sensors integrated into this system. However, SDI-12 differs from typical UART communication in that its bit values are inverted and requires strict timing. To address this, the project leveraged the RP2040's UART capabilities in conjunction with an RS485 transceiver. The RS485 transceiver is suitable for this application due to its robust line-driving capabilities, noise immunity, and compatibility with inverted logic levels. By using only the non-inverting line of the differential pair (data line B), and setting the voltage reference at 2.5V, the design effectively maps the inverted SDI-12 signals to conventional UART high and low values, allowing reliable sensor communication.

Power Supply and Voltage Regulation

Due to the diversity of communication protocols and components used in the system (such as ADC for the load cell, SDI-12 for sensors, and I2C for the DAC—multiple), the following voltage levels were required on the PCB: 12V, 5V, and 3.3V. Each voltage level was supplied through dedicated regulators to ensure stable power delivery to each module. The careful placement of decoupling capacitors was critical to minimize voltage ripple and noise, ensuring stable operation of the entire system.

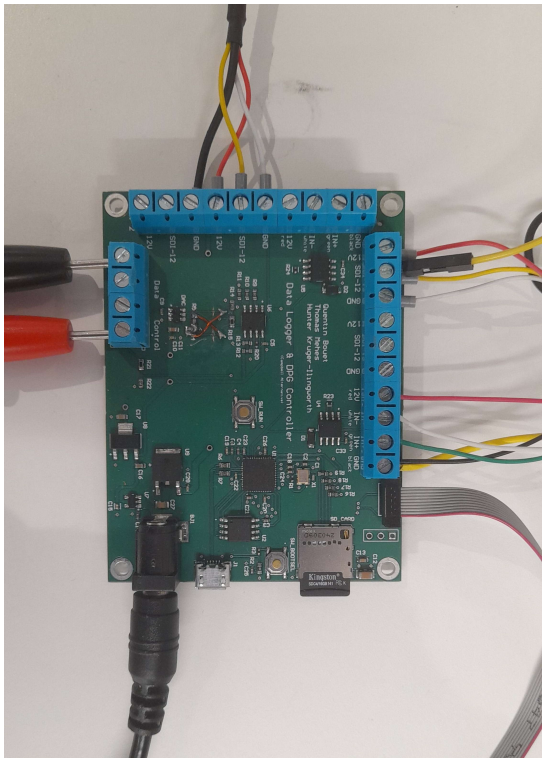

Fig. 2. Photo of Board

Data Logging and Storage

An SD card module was added to the schematic, utilising the SPI interface as defined in the RP2040 datasheet. This module enables reliable data storage and allows easy access to logged data for further analysis. The SPI bus operates at a clock speed of 1 MHz, which is suitable for reliable data transfer with the SD card.

DAC Integration

Initially, a solution consisting of Pulse Width Modulation (PWM) along with a low pass filter was considered to convert to generate the required analogue signal for the dew point generator. However, this method proved unreliable and inaccurate when tested with a Raspberry Pi Pico. As a result, the MCP4716 DAC (10 bits) was integrated to provide fine-grained control of the dew point generator. Proper bypass capacitors were placed near the V_{DD} pin to minimize induced noise and ensure stable operation. The DAC was designed to communicate with the RP2040 over the I2C bus at a standard mode speed of 100 kHz.

Communication Protocols and OSI Model Layers

In designing the embedded system, several communication protocols were implemented to interface with various sensors and peripherals. Understanding how these protocols map onto the OSI model layers, particularly the physical and data link layers, is crucial for ensuring reliable and efficient data transmission. This section explains the OSI model implementation for each protocol used: SDI-12/UART/RS485, SPI, I2C, and analog inputs and outputs.

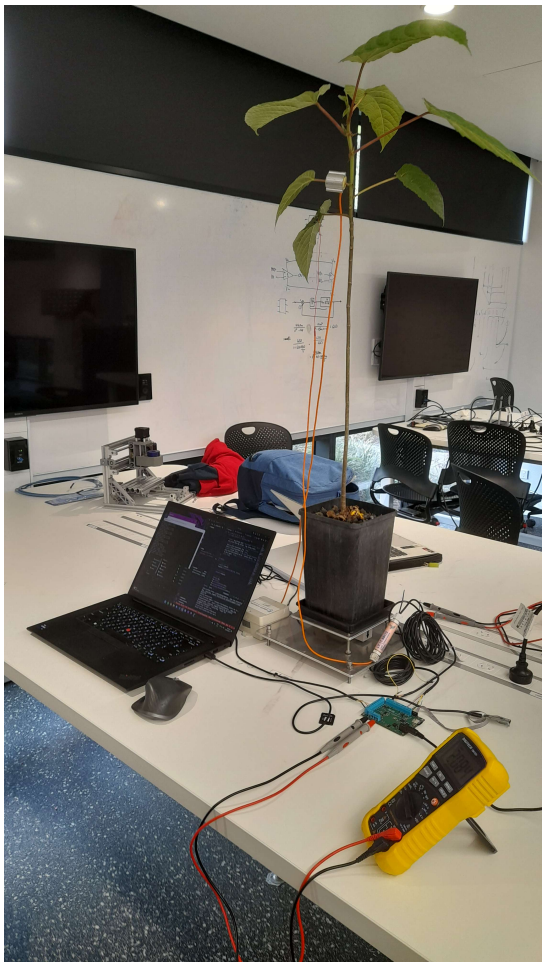

Fig. 3. Photo of System Setup

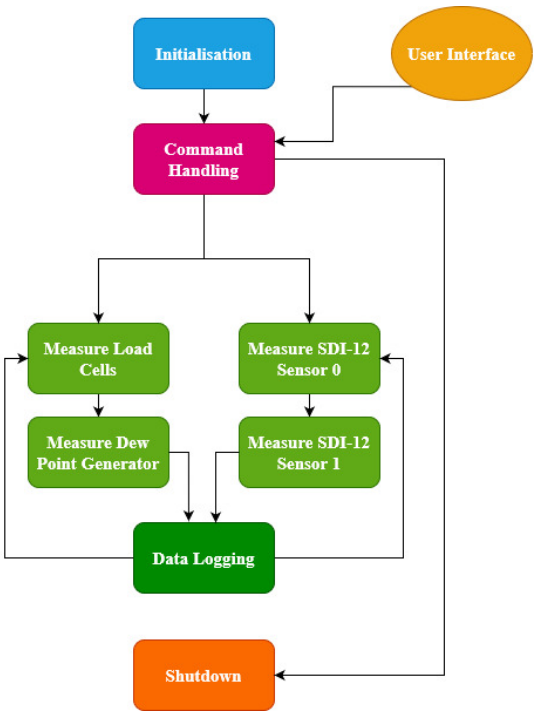

Fig. 4. Flowchart of Program

The program sequentially takes measurements from each SDI-12 sensor while simultaneously collecting data from faster sensors, such as load cells and the dew point generator, during the waiting periods for SDI-12 responses. Instead of using interrupts, the program relies on flag-based polling to gather data. Since the system is monitoring a living plant, where rapid changes are not expected, precise timing is not critical. The following diagram (fig. 4) illustrates the process:

SDI-12/UART/RS485 Communication

Physical Layer

SDI-12 communication is managed using UART1 on the RP2040 microcontroller, with specific GPIO pins defined for transmission (TX), reception (RX), and data line drive enablement. The SDI-12 protocol operates over a single data line with negative logic levels: logic '0' is high voltage; logic '1' is low voltage. The RS485 transceiver adapts the UART signals to meet these requirements.

The RS485 transceiver is suitable because:

- **UART Compatibility**: It interfaces easily with UART and supports SDI-12's 1200 baud rate.
- **Noise Immunity**: Differential signaling enhances noise immunity for reliable long-distance communication.
- **Line Driving Capability**: Drives long, high-capacitance cables, maintaining signal integrity.
- **Inverted Logic Handling**: Accommodates the inverted logic levels of SDI-12 devices.

By using only the non-inverting line and proper voltage references, we adapted UART to meet SDI-12's physical layer requirements.

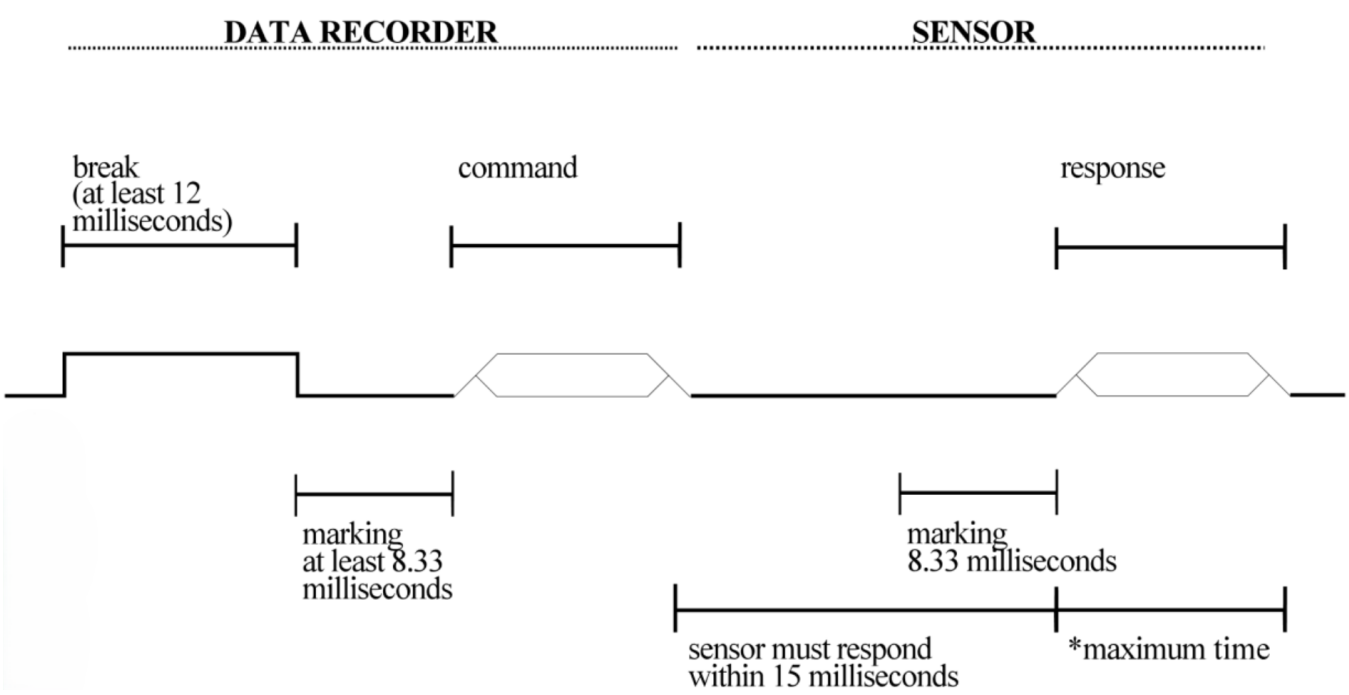

Fig. 5. SDI-12 timing from SDI-12 Support Group (2019)

Data Link Layer

The SDI-12 protocol defines communication between the data recorder (master) and sensors (slaves) at 1200 baud, using 7 data bits, even parity, and one stop bit. It specifies ASCII commands for initiating measurements, requesting data, and managing sensor addresses.

In the current implementation, the system communicates with two SDI-12 sensors with hardcoded addresses. While this simplifies software development, it limits scalability and requires prior knowledge of sensor addresses. SDI-12 supports up to ten sensors on a single data line, each with a unique address. Implementing a dynamic discovery process using the "Address Query" command (?!) would allow communication with multiple sensors without prior address knowledge, enhancing extensibility.

Application Layer

Custom drivers were developed for each sensor, adhering to the SDI-12 protocol. Specific issues with the byte framing of the protocol were addressed by using the `uart_set_format()` function to configure the communication format correctly, allowing the RP2040 to receive inverted data without additional circuitry. The `uart_break()` function sends the required break signal, and `uart_read_stuff()` reads sensor responses, which are then parsed to extract data. The timing of the SDI-12 protocol is shown in fig. 5.

SPI Communication (SD Card)

Physical Layer

The SPI protocol is used for communication with the SD card module. SPI is a synchronous serial communication interface that operates in full-duplex mode. It uses four signals:

- MOSI (Master Out Slave In): Transfers data from the master to the slave.
- MISO (Master In Slave Out): Transfers data from the slave to the master.
- SCK (Serial Clock): Synchronises data transmission between the master and slave.
- SS (Slave Select): Enables the slave device for communication.

In this system, the SPI bus operates at a clock speed of 1 MHz, suitable for reliable data transfer with the SD card.

Data Link Layer

At the data link layer, the SPI protocol exchanges data bytes between the microcontroller and the SD card. It does not define a specific frame format or error-checking mechanism, relying on higher-level protocols for these functions. The FatFs file system library is employed to handle file operations on the SD card. FatFs provides functions for reading, writing, and managing files, ensuring data integrity and proper formatting according to the FAT file system standards.

Application Layer

The FatFs API was integrated to enable read/write operations on the SD card, using the SPI protocol. This allowed for structured data logging and easy retrieval of information. Functions were implemented to write data in CSV format, facilitating compatibility with data analysis tools.

I2C Communication (DAC)

Physical Layer

The I2C protocol is used for communication with the MCP4716 DAC. I2C is a synchronous, multi-master, multi-slave, single-ended serial communication bus. It uses two bidirectional open-drain lines:

- SDA (Serial Data Line): Transfers data between devices.
- SCL (Serial Clock Line): Provides the clock signal for synchronisation.

Pull-up resistors are used on both lines to maintain the high logic level when the bus is idle. In this system, the I2C bus operates at a standard mode speed of 100 kHz.

Data Link Layer

At the data link layer, the I2C protocol handles addressing and data transfer between the master (RP2040 microcontroller) and the slave device (DAC). Each device on the I2C bus has a unique 7-bit address. The protocol defines start and stop conditions, acknowledgments, and data byte transfers. The software includes drivers to configure the DAC's settings, such as reference voltage and gain, and to send digital values that the DAC converts into analog voltages. This allows precise control over the dew point generator.

Analogue Inputs and Outputs

Physical Layer

Analog inputs and outputs interface with devices like the load cell (MT-603) and the dew point generator (LI-610). The load cell produces a voltage proportional to the applied weight, connected to the RP2040's ADC pins. An instrumentation amplifier (INA826) amplifies the small signal from the load cell. The dew point generator accepts a 0–5 V analog input for control, provided by the DAC's output.

Data Link Layer

While analog signals lack a formal data link layer, signal conditioning and conversion serve similar functions. For analog inputs, the load cell signal is amplified and filtered before ADC digitization. For analog outputs, the DAC converts digital values into precise voltages to control the dew point generator's settings.

Application Layer

A software interface controls the dew point generator via the I2C protocol communicating with the DAC (MCP4716). This allows precise voltage adjustments, with tests confirming accurate outputs based on command inputs. I2C communication operates in standard mode at 100 kHz, compatible with the DAC.

Selecting appropriate communication interfaces and voltage levels, along with robust software integration, enables the system to operate seamlessly and efficiently, providing a reliable solution for monitoring and controlling environmental parameters.

Command-line Interface

A command-line interface was implemented to allow the user to interact with the system. A terminal driver was developped to handle user input and execute commands and manages a buffer for multi-character commands. A command in this context is represented as a data structure called `Command`, which encapsulates the command type and an optional floating-point argument. The 'Command' struct contains:

- the type of command issued by the user (e.g., `help`, `set_voltage`, `get_data`, `shutdown`)
- an optional argument for commands that require additional information, such as a voltage value for `set_voltage`

This class provides methods for:

- Handling input character by character.
- Parsing and executing user commands by interpreting the `Command` structure.
- Resetting and managing the internal command buffer used for building commands from user input.

Discussion

This section will discuss the implications of the design choices, critically assessing their effectiveness and potential areas for improvement.

The main issue will be having the interrupts not "collide" with each other since the low baud rate increases the probability that other things will interrupt the middle of a communication session with one of the SDI-12 sensors.

Technical Challenges and Solutions

SDI-12 Sensor Communication Issues

During the testing of the SDI-12 sensors, initial attempts to communicate using RS485 transceivers failed due to improper framing of byte streams. The strict timing requirements and inverted logic levels of the SDI-12 protocol made it difficult to achieve reliable communication using standard UART settings. This issue was addressed by applying the `uart_set_format()` function, which allowed the RP2040 to handle the SDI-12 protocol's timing and framing more accurately.

Additionally, a `custom is_timed_out()` function was created to automate character reception without using traditional sleep methods, ensuring precise byte timing and avoiding data loss during transmission. The RS485 transceiver's ability to handle inverted logic and differential signaling enhanced noise immunity and improved communication reliability over long distances.

Load Cell Signal Scaling and Stability

The MT-603 load cell exhibited a significant dead zone and provided inconsistent readings due to mechanical vibrations in the experimental setup. An instrumentation amplifier was used to amplify the load cell's signal, improving sensitivity. However, environmental noise and mechanical oscillations still affected the readings.

Software-based noise reduction techniques, such as averaging past data points using an exponential moving average, were implemented to mitigate the effect of oscillations. This software solution was preferred over hardware filtering due to the constraints of the existing apparatus.

I2C Bus and DAC Communication Issues

During the process of integrating the DAC, software implementations from GitHub were tested, but scanning the I2C bus showed no devices detected. This pointed to a hardware issue. Upon further inspection, it was discovered that the SDA and SCL pins were swapped on the RP2040, and the pull-up resistors for the DAC were incorrectly sized.

To address this, the traces were manually cut and re-soldered to correct the pin connections. Smaller pull-up resistors were also added to compensate for the track capacitance, which had caused the rising edges of the clock signal to appear more triangular than square. After these modifications, the I2C bus functioned correctly, and basic software confirmed the DAC's operation. This highlighted the importance of thoroughly reviewing the final PCB layout before submission, as the initial errors could have been avoided.

Recommendations

A more extensible system would ask all the sdi12 sensors on the data line what their address is and dynamically get data from all of them regardless of how many sensors there are. The hardware design is highly extensible (screw in terminals allow you to plug in however many you want) but the software would need to be developed more to fully realise this extensibility (mention specific commands that could be used to get all the sensors address)

A potential solution is to implement an SDI-12 command queue, which would manage sensor requests efficiently. Alternatively, a system similar to real-time operating systems (RTOS) could be used, where tasks are prioritized and managed with mechanisms like task prioritization and mutexes. This approach would ensure smoother task allocation, preventing conflicts and improving response handling for both fast and slow sensors.

Based on the design process and challenges encountered, several recommendations can be made for future engineers who might revise this work:

1. Implement Dynamic Sensor Discovery: To enhance the system's scalability and flexibility, incorporate the SDI-12 "Address Query" command (?!) to dynamically discover all connected sensors. This would eliminate the need for hardcoded sensor addresses and allow the system to support any number of SDI-12 sensors without software modifications.

2. Develop Robust Error Handling: Improve the software to handle communication errors, sensor timeouts, and unexpected responses. Implementing retries, acknowledgments, and exception handling will make the system more reliable in field conditions.

3. Optimise Hardware Design for Extensibility: Ensure that hardware interfaces, such as screw terminals and connectors, are designed to accommodate future expansions. Consider modular designs that allow for easy addition or replacement of components.

4. Thorough PCB Layout Review: Implement a rigorous design review process before finalising the PCB layout. This includes checking pin assignments, signal integrity, and component placements to prevent issues like swapped pins or incorrect resistor sizing.

5. Enhanced Signal Conditioning: For analogue inputs like the load cell, consider implementing hardware filtering and shielding to reduce noise and improve signal stability. This might include adding low-pass filters or using differential signal processing techniques.

6. Documentation and Standardisation: Maintain comprehensive documentation of both hardware and software components. Adhering to coding standards and providing clear comments will facilitate future development and maintenance.

7. Power Management: Explore power-saving features and strategies, especially if the system is to be deployed in remote locations where power availability is limited.

By addressing these areas, future iterations of the system can achieve greater robustness, scalability, and ease of use, ultimately enhancing its value for environmental research applications.

Conclusion

This section will summarize the project's outcomes, reflecting on the objectives achieved and providing final thoughts.

Appendix

References

SDI-12 Support Group, *SDI-12: A Serial-Digital Interface Standard for Microprocessor-Based Sensors*, SDI-12 Support Group, River Heights, Utah, Jan. 2019, version 1.4. [Online]. Available: http://www.sdi-12.org