GLASGOW CALEDONIAN UNIVERSITY

MEng Group Research Project

MMH723842-24-AB-GLAS

Design and Implementation of a Photodiode Array-Based Analogue 2D Sun Sensor

word count: xxx

by Zac McCaffery, Alexandru Belea,
Sebastian Alexander, William Kong, Nassor Salim,

Date: April 12, 2025

Contents

List of Figures

Abstract

add abstract here

1 Acknowledgements

We would like to express our sincere gratitude to our supervisors, Dr. Roberto Ramirez-Iniguez and Geraint Bevan, for their invaluable guidance and unwavering support throughout this project.

Our appreciation extends to the 3rd Floor EEE Lab Technicians and Dr. Carlos Gamio-Roffe, whose technical expertise and assistance were instrumental in the successful construction of the prototype.

We are particularly grateful to the European Project Semester RED Team members—Nikolay Ivanov Shopov, Stef Hannisse, and Samridhi Gupta—for generously allowing the use of their Renewable Energy Demonstrator as a testbench. Their contribution provided an ideal platform for positioning light sources during the testing phase of this project.

2 Introduction

2.1 Problem Statement

With the ever-increasing commercialization of the space and satellite industry there is a growing need for a cost-effective method of attitude tracking for smaller satellite missions of such as CubeSat as these missions are purpose built for very specific objectives. Whilst the larger commercial satellite missions make use of expensive digital camera systems for tracking purposes, this is not feasible for much smaller CubeSat setups. CubeSats are defined from 1 unit to 12 – where 1U is a 10x10x10 cm satellite. Consequently, there is a demand for a cost-effective and easily implementable attitude tracking system that can provide accurate measurements for CubeSat missions, such as a Position Sensitive Detector (PSD) using photodiodes.

2.2 Aim of the Project

"To investigate and develop a cost-effective and reliable sun sensing solution suitable for Low Earth Orbit (LEO) nanosatellite attitude determination."

2.3 Objectives of the Project

To investigate the design of a sun sensing system for nanosatellites, used in orientation determination, through detection of its relative position to the sun using analogue sensors located on the satellite's body. Our goal is to create a system which balances cost-effectiveness and simplicity. To achieve this, we will create a software model of the analogue sensor(s) to simulate the system's ability to track the sun from various angles in orbit. After which, we aim to build a physical prototype and use a movable light source to simulate the sun's movement, allowing comparison between the real sensor's performance against our simulations. Although the physical prototype will be built using non-space-grade materials, one of the objectives is to look at and analyse materials required for building a space-grade PCB and sensor. For this step, the Mechanical side of the team will perform Printed circuit board (PCB) and aperture device finite analysis using ANSYS to determine resilience to environmental factors such as stress and thermal simulation. The application of signal processing will be explored to provide usable data, filter out

noise, and improve the system's accuracy. This approach aims to develop a cost-effective and reliable, in-house sun sensing solution specifically for nanosatellites operating in Low Earth Orbit. Major Objective points:

- **Conduct literature review:**
  - Analyse existing research on sun sensing technologies, with a focus on PSD-based analogue sensors and their applications in nanosatellites.
  - Identify current challenges, best practices, and advancements in attitude determination in Low Earth Orbit. Use these insights to guide the design and optimisation of the proposed sun sensing system.
- **Develop software model:**
  - Simulate the performance of the PSD-based analogue sun sensor in tracking the sun's position from various angles in Low Earth Orbit.
- **Design and fabrication of physical prototype:**
  - Integrate analogue sun sensor components, test and validate its performance under controlled conditions.
- **Compare simulated and experimental results:**
  - Establish evaluation methodology between simulated and experimental test results to ensure that topology evaluation is applicable.
- **Optimise sensor topology:**
  - Research and evaluate various configurations of analogue sun sensing systems to maximise sun detection accuracy and minimise blind spots.
- **Investigate environmental factors:**
  - Evaluate the material requirements of the PCB and aperture device.
- **Implement signal processing algorithms:**
  - Investigate the filtering of noise to enhance the signal-to-noise ratio and otherwise ensure the acquisition of usable data for accurate sun position determination.
  - Implement data handling which optimises scanning rates and efficiently processes the analogue signal data for real-time attitude determination.
- **Document results and overall cost-effectiveness:**
  - Develop criteria for final evaluation of sun sensing systems, on which to base the final presentation of project findings.

3 LiteratureReview

3.1 CubeSat Design

Puig-Suari, Turner and Ahlgren published an IEEE paper in 2001 with the help of their students at California Polytechnic State University exploring a need for micro satellites for use by universities in an ever-expanding space programme. They provide as a solution a standard satellite form-factor that will bring down the cost of both manufacture and deployment of satellites by smaller entities: the CubeSat. The paper identifies a key component for the success of this form factor a need for a standard CubeSat deployer mechanism which can deploy several satellites safely and develop such a platform, called Poly Picosatellite Orbital Deployer or P-POD. They point out the need and provide microsatellite size and shape of the CubeSat form factor [1]. Sai balaji et al. performed a study using MATLAB simulation of several attitude control algorithms to look at the ability to control a CubeSat of size 1U. They also simulated sensors such as sun sensors, magnetometer, and gyroscope. They concluded that it is possible to operate the satellite using a magnetorquer type actuator and an array of mathematical models and algorithms: it would take 2000 seconds for a 1U satellite to stabilize at 505km, 98° degree attitude in orbit with the methods utilized by them [2]. Incentivised by the rapidly increasing use of LEO, Lopez-Calle and Franco perform a quantitative comparative study on the catastrophic failure of CubeSats and Nanosats from radiation exposure due to the harsh environment of space versus failure due to collisions in the increasingly busy Low Earth Orbit (LEO). The authors concluded that while sustained damage and damage protection from radiation exposure used to be and currently still is the most crucial factor in protecting LEO microsatellites, increasingly the risk of debris collisions is becoming more important and will become the most important in the following 50 to 70 years. The authors conclude that microsatellite designers need to move their focus more towards defence from debris impacts as these, even if not resulting in catastrophic failure of the satellite, they will impact the attitude of the satellite [3].

4 Methodology

4.1 Python Simulation tbc

4.2 Prototype Development

4.2.1 Lifecycle

This section provides an overview of the Prototype Development Lifecycle.

Conceptualization and Requirements Definition

- The prototype must have four photodiodes in an xy pattern with respective circuitry required to output 0-5 Volts that will be read by an Arduino based Data Acquisition System (DAQ). The circuit must be able to react to light intensity changes, however the change will be at low frequency (below 1Hz) as a satellite attitude is considered to change only gradually.
- While the prototype may not have a high accuracy, it is hoped that it will be enough to measure light position changes.
- The prototype within the scope of this paper will show the ability to detect the position of light at normal room conditions, therefore it does not need to withstand temperature changes or radiation that a final product would require if deployed in space.
- Interface requirements: the prototype electrical output needs to be compatible with the Arduino Analog to Digital Converter (ADC) input. Therefore, the signal shall not go below 0 Volts or exeed 5 volts.
- Size and weight are not of high importance, but the device must fit in the testing equipment, which is the Renewable Energy Demonstrator arch. Preferably a height not higher than 5cm.

Theoretical Design

- Research photodiode technology options and selection criteria

- Model sun sensor geometry and aperture design
- Determine optimal photodiode placement for coverage and accuracy
- Develop mathematical models for sun vector determination
- Simulate sensor performance under various lighting conditions

Preliminary Design

- Create detailed electrical schematics
- Design photodiode array configuration
- Engineer aperture design and mask pattern
- Develop analog front-end circuitry
- Design signal conditioning and processing circuits
- Outline firmware architecture and algorithms

Component Selection and Procurement

- Select appropriate photodiodes (spectral response, sensitivity)
- Choose microcontroller/processor
- Source analog-to-digital converters
- Identify appropriate materials for aperture and housing
- Procure test equipment for validation

Breadboard Testing

- Assemble basic circuit on breadboard
- Test photodiode response characteristics
- Verify analog front-end performance
- Validate signal processing approach
- Identify design weaknesses and optimization opportunities

First Prototype Development

- Design printed circuit board (PCB)
- Manufacture PCB
- Design and fabricate aperture mask
- Develop housing/enclosure
- Assemble prototype components
- Write initial firmware implementation

Initial Testing and Characterization

- Conduct functional testing
- Measure photodiode response curves
- Characterize sun angle determination accuracy
- Test temperature sensitivity
- Evaluate power consumption
- Assess signal-to-noise ratio

Design Refinement

- Analyze test results
- Modify aperture design if needed
- Optimize photodiode configuration
- Update signal processing algorithms
- Refine PCB layout
- Improve firmware algorithms

Second Prototype Development

- Implement design improvements
- Manufacture revised PCB
- Fabricate improved aperture

- Enhance housing design
- Update firmware with optimized algorithms
- Assemble refined prototype

Comprehensive Testing

- Laboratory performance testing (angular accuracy, resolution)
- Environmental testing (thermal cycling, vibration)
- Radiation testing (if applicable for space applications)
- Interface compatibility testing
- Long-term stability assessment

Validation and Calibration

- Develop calibration procedures
- Create calibration fixtures
- Perform sensor calibration
- Document calibration coefficients
- Validate sensor performance against requirements

Documentation and Production Readiness

- Create detailed technical specifications
- Document calibration procedures
- Prepare assembly instructions
- Write user manual/interface control document
- Develop acceptance test procedures

Pre-production Prototype

- Build small batch of pre-production units
- Conduct acceptance testing
- Verify production processes
- Validate consistency between units
- Finalize design for production

Technology Transfer to Production

- Document manufacturing processes
- Train production personnel
- Establish quality control procedures
- Define production testing requirements
- Prepare for volume manufacturing

4.3 System Design Overview

This section provides an overview of the System Design Overview.

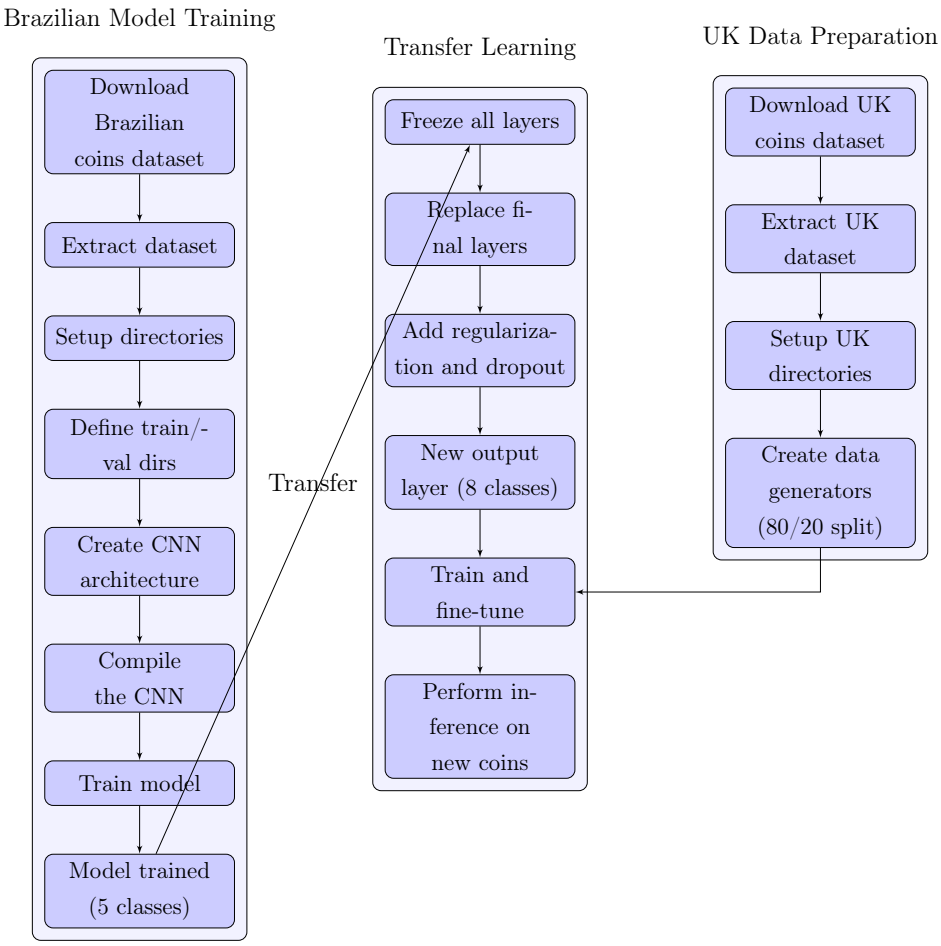

Figure 4.1: System Design Overview Flowchart

4.3.1 Functional Requirements

4.3.2 Design Approach

4.3.3 System Architecture

As shown in Figure 4.1 the system architecture consists of various components.

```
# Your code here
```

Listing 4.1: System Architecture Code Example

4.4 Sensor Array Development

This section provides an overview of the Sensor Array Development.

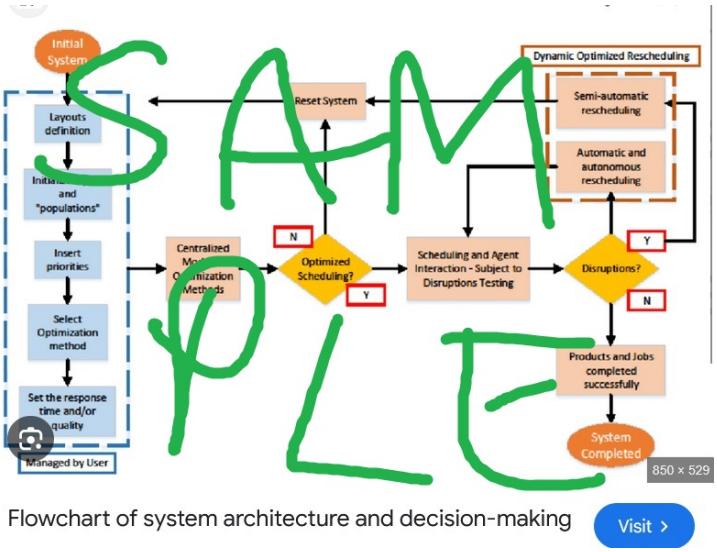

Figure 4.2: System Architecture Diagram

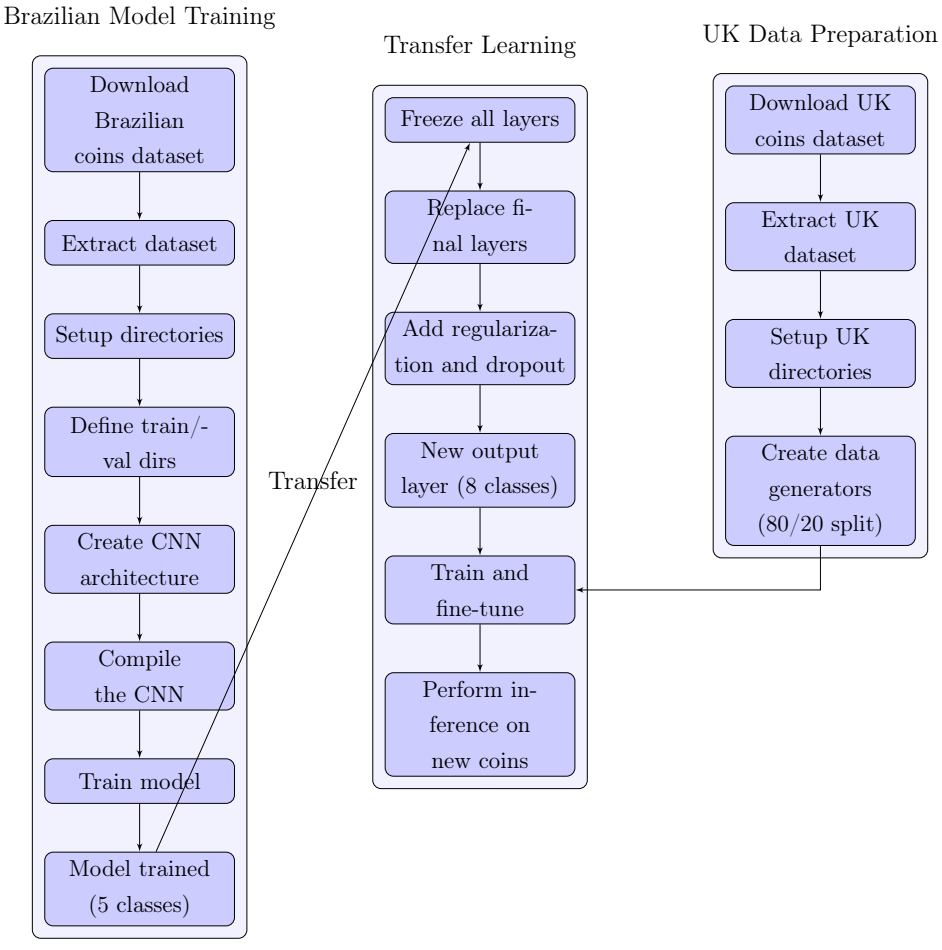

Figure 4.3: System Design Overview Flowchart

4.4.1 Functional Requirements

4.4.2 Design Approach

4.4.3 System Architecture

As shown in Figure 4.3 the system architecture consists of various components.

```
# Your code here
```

Listing 4.2: System Architecture Code Example

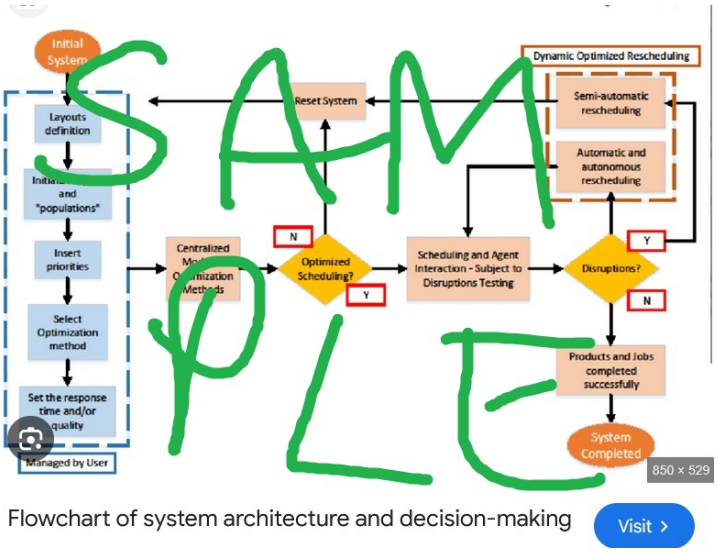

Figure 4.4: System Architecture Diagram

4.5 Signal Conditioning Circuitry

This section provides an overview of the Signal Conditioning Circuitry.

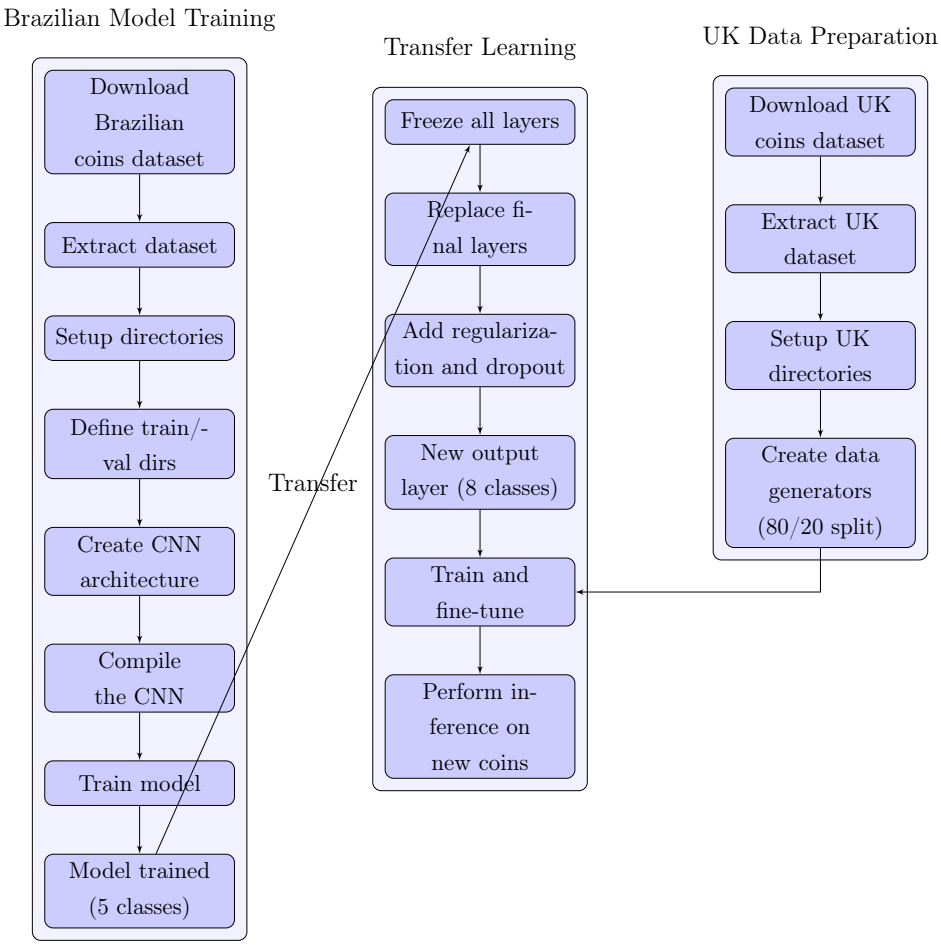

Figure 4.5: System Design Overview Flowchart

4.5.1 Functional Requirements

4.5.2 Design Approach

4.5.3 System Architecture

As shown in Figure 4.5 the system architecture consists of various components.

```
# Your code here
```

Listing 4.3: System Architecture Code Example

4.6 Enclosure Design And Fabrication

This section provides an overview of the Enclosure Design And Fabrication.

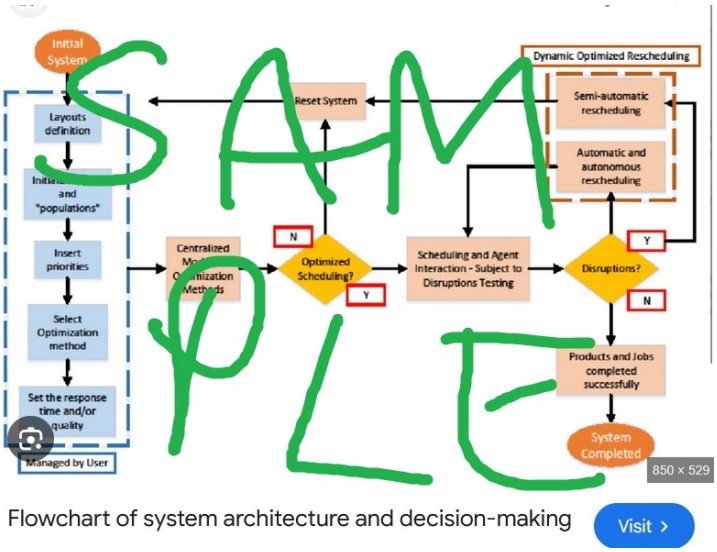

Figure 4.6: System Architecture Diagram

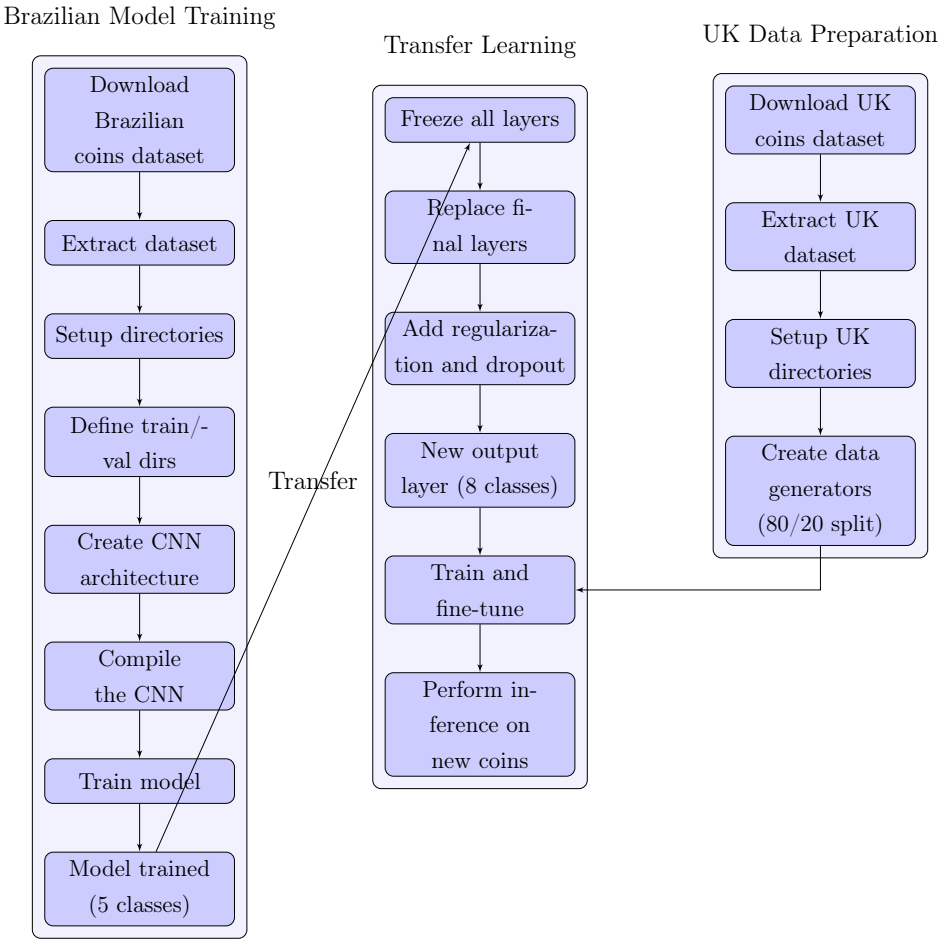

Figure 4.7: System Design Overview Flowchart

4.6.1 Functional Requirements

4.6.2 Design Approach

4.6.3 System Architecture

As shown in Figure 4.7 the system architecture consists of various components.

```
# Your code here
```

Listing 4.4: System Architecture Code Example

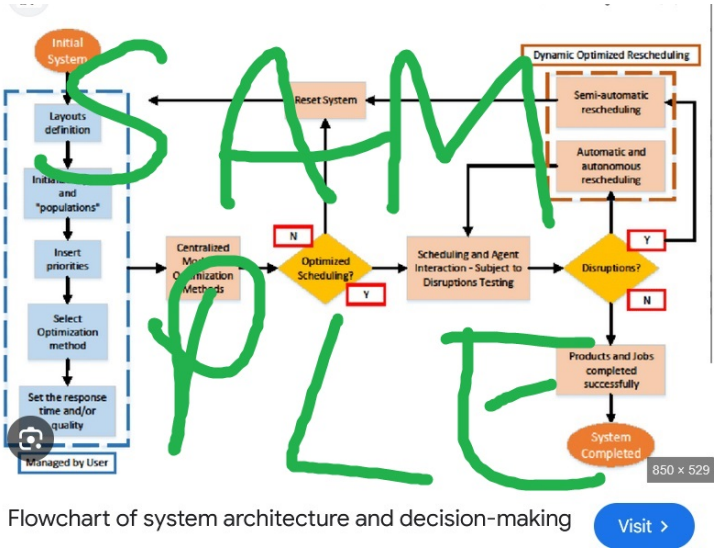

Figure 4.8: System Architecture Diagram

4.7 Data Acquisition System

4.7.1 Functional Requirements

The output signal from the photodiode array is required to be converted to digital form for post processing. This requirement is filled by designing a Digital Acquisition System (DAQ) capable of recording the signal from the four photodiode circuits simultaneously. The choice of design was conceived by analyzing the analog signal and determining some basic requirements of the Analog to Digital Converter (ADC) the DAQ must possess.

Analog Signal Characteristics

- The signal is four channel, one per photodiode, and between 0 and 5 Volts, as the TIA and post amplification was designed specifically for this output.
- Close to DC frequency, i.e., static in nature, due to light intensity remaining static under most tests. One test is performed at 0.2Hz, which is still still very low

frequency, with the light completing a semicircular arc once in 130 seconds (26 positions of 5 seconds each).

- Later in testing it was found that the signal is impacted by interference of 400mVpp at a frequency fluctuating from 160kHz to 180kHz from the RED testbench power supply, as pictured in Figure 4.9.

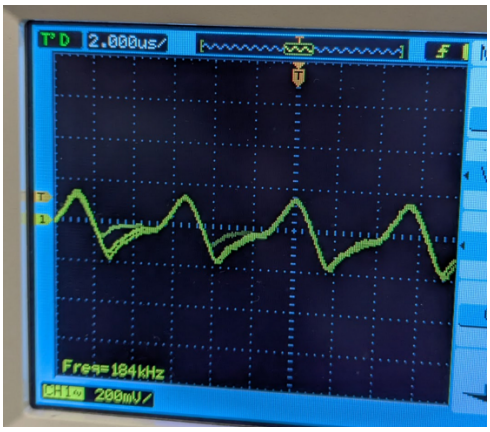

Figure 4.9: Signal Noise Analysis, oscilloscope AC coupled

CSV Data Structure and Format Specification is as follows:

The output of the DAQ is to be saved in Comma Separated Values (CSV) file format, with columns as follows: Sample, Time(ms), A0(V), A1(V), A2(V), A3(V). This allows for easy post-processing and plotting.

4.7.2 Design Approach

The characteristics of the signal being low frequency, combined with the requirement to read all four signals simultanously and in sync, meant two things: the Sampling Rate could be quite low due to Nyquist theorem telling us that the sampling rate must be at least twice the frequency of the signal being sampled, in order to maintain the original signal without aliasing [4, p. 146]. Therefore a low performing ADC is acceptable for a signal changing at under 1Hz. And secondly, the DAQ must support sampling from at least four analog inputs. These requirements meant that a cheap Arduino based DAQ could fit perfectly the needs of the project: it is powered by the Atmega328P which has an included ADC of 15 ksps [5, p.205]. And the Arduino Nano has four analog inputs.

Arduino Programming

The Arduino-based DAQ will require both a C++ program written for the Arduino itself, as well as a program or script on the PC receiving the digitized signal, this is because

the Arduino lacks both the memory requirements and capability to store the recorded digitized signal to some internal memory.

The Arduino C++ Program must be able to listen to commands from the user on the PC receiving, start a recording, and immediatly transmit to the PC over serial communication.

4.7.3 Technical Specifications

Arduino Code

Introduce Code

```
recordingDuration = 5000 // for how long to record in milliseconds

minSampleInterval = 2    // control how fast to sample to avoid
                         // relying on Arduino performance
// Initialize serial communication
// Initialize analog inputs
// Setup ADC

// Infrom PC listening on Serial Connection: Arduino is ready to
 record
Serial.print("Arduino_DAQ_Ready")

void loop(){
  //listen for command from PC script:
    String command = Serial.read()
  //set system state
  if (command == "START"){
    // Send header of csv
    Serial.println("Sample,Time(ms),A0(V),A1(V),A2(V),A3(V)")
    // keep track of system state
    state = recording
    // keep time
    startTime = currentTime()
    // send confirmation
    Serial.println("recording in progress")
    }
  // check if recording
  if (state == recording){
    // check if within recording period
    currentTime = currentTime()
    elapsedTime = currentTime - startTime
    if(currentTime <= recordingDuration){
      // Also check not recording too fast
      if(currentTime - lastSampleTime >= minSampleInterval){
```

```
          sampleCount++;
          // Start each row with sample count and time of sample
          String currentCSVrow = String(sampleCount) , string(
   elapsedTime)
          // Multiplex through all analog inputs
          for (int = 0;i<4;i++){
            //read raw values
            rawValue = analogRead(analogInputs[i]);
            // compute real value
            voltage = rawValue * 5/1023
            // add value to current row to send
            currentCSVrow += String(voltage)
            }
          // Send completed row to PC
          Serial.println(currentCSVrow)
          }
        }
      }
  // end recording
  else{
    state = notRecording
    // tell PC recording finished
    Serial.println("Recording_finished")
    }
  }

```

Listing 4.5: Arduino DAQ PseudoCode

Sampling Rate Several factors restrict the sampling rate:

Sample Interval Setting The most direct limitation is the `sampleInterval` constant set to 2ms in the code, which means samples are taken no more frequently than every 2 milliseconds (500 Hz theoretical maximum).

ADC Prescaler Configuration The ADC prescaler is set to 16 (from the default of 128) with this line:

```
ADCSRA = (ADCSRA & 0xF8) | 0x04;
```

This increases the ADC clock to 16MHz/16 = 1MHz. With each conversion taking 13 ADC clock cycles, the theoretical maximum sampling rate is about 76.9kHz for a single channel.

Multiple Channel Reading Since the system reads from 4 analog inputs sequentially, the effective per-channel sampling rate is reduced by approximately a factor of 4.

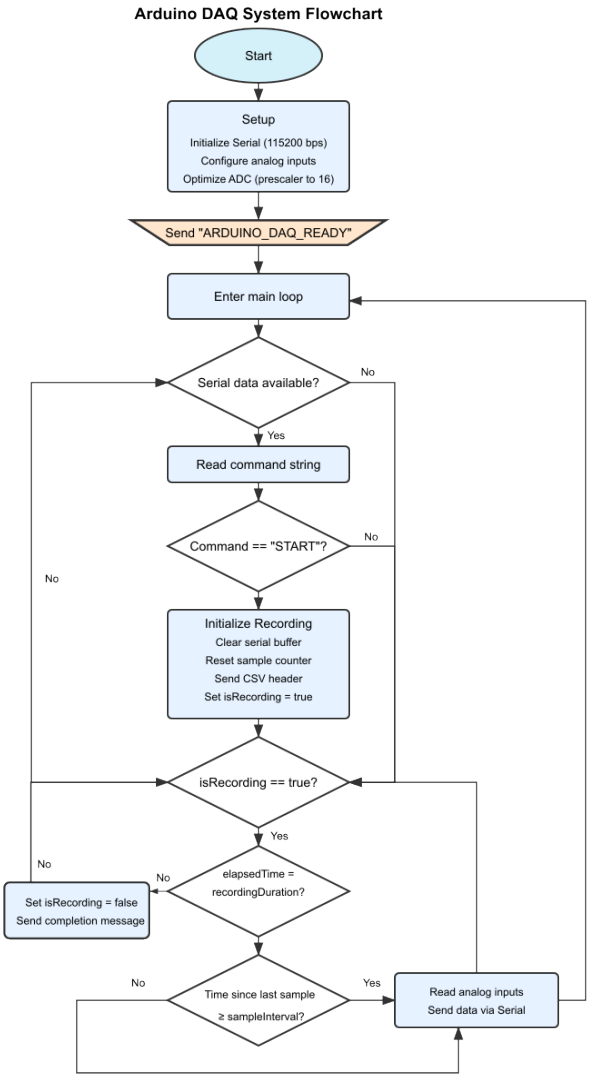

Figure 4.10: Flowchart of C++ code on Arduino DAQ

Serial Transmission Overhead Each sample requires formatting and sending data over serial:

```
String dataString = String(sampleCount) + "," + String(elapsedTime);
// ... format and add voltage values ...
Serial.println(dataString);
```

This string creation and serial transmission takes considerable time.

Serial Baud Rate The code uses 115200 bps, which limits how quickly data can be transmitted. Each sample in this format might be around 30-40 bytes, which means ~3000-3800 samples/second theoretical maximum throughput.

String Operations The use of the Arduino `String` class is memory-intensive and can cause fragmentation over time, potentially causing slowdowns.

Loop Cycle Time Other operations in the main loop consume processing time.

The dominant limiting factor in this implementation is likely the combination of the explicit 2ms sample interval and the serial communication overhead. Higher sampling rates would require optimizing the data transmission format, possibly using binary rather than text formatting, and reducing the sample interval.

4.8 Renewable Energy Demonstrator Testbench

For testing the capability of the Sun Sensor to correctly detect the location of the light source, a test bench was required that could reliably place the location of the light at a precise location repeatedly. For this purpose we used a project built by our colleagues in the European Project Semester year 2021/22 who createad just such a device intended for demonstrating renewable energy creation live [6]. Their device was able to demonstrate the energy levels created by a Photovoltaic (PV) cell by light emited at different angles. The light emission would change location based on time of day and the PV cell readings would show the difference in energy. Futher the PV cell was controllable by a joystick to point the PV Cell at the optimum angle for the highest energy capture. For our project, the arch and LED strip were used for outputing light from different angles.

Analysis of High Frequency Noise in AC-DC Power Supply

Interference structure of around 170kHz with 400mV peak-to-peak was detected on the signal being received while the RED testbench was on as shown in Figure 4.9. This noise could be generated by several factors in the AC power supply used by the RED testbench:

1. **Switching frequency harmonics** — If it's a switch-mode power supply (SMPS), the fundamental switching frequency or its harmonics might be causing the noise. Many SMPS operate in the 50–200,kHz range.

2. **Poor filtering** — Inadequate output filtering (insufficient capacitance or poor quality capacitors) can allow switching noise to appear on the output.

3. **Improper design of magnetics** — Issues with the transformer or inductor design could cause ringing or oscillations.

4. **Resonance in the circuit** — Parasitic capacitance and inductance forming a resonant circuit at around 170,kHz.

5. **Control loop instability** — PWM controller instability can cause oscillations.

6. **Ground loops or poor PCB layout** — Improper grounding or PCB layout can create noise paths. [7]

To avoid spending time diagnosing and trying to repair the testbench, an easier solution was reached: performing digital filtering of the acquired signal in post processing. Due to the signal of interest being close to DC - frequencies much lower than 1Hz, and the noise being high frequency, around 175kHz, a simple digital Butterworth filter with a cutoff frequency at around 1-2 Herz was found to be a good solution.

The only remaining issue was that this noise would sometimes trigger the internal components of the testbench, unintentionally triggering the button press from the control interface that was changing the light position, but it happened so rare that it was not a major concern.

5 Software Model

5.1 Introduction

A Python model was constructed to provide a simulation of the movement and intersection of rays from a movable source to evaluate sensor performance and compare these results with practical experiments. The model allows for a number of configurable parameters:

- The trajectory of the light source 3D space, which moves in configurable discrete increments.
- The placement of any number of sensors and apertures, including their dimensions.
- The form of the output data, including as a static, or animated graphic.

Affording flexibility for the model to simulate any sensor topology under a variety of conditions.

5.2 Theory and Concept

The system is modelled in 3D space, consisting of planes and lines. Each line is defined by vectors representing position and normal direction, $\vec{A} = (a, b, c)$ and $\vec{u} = (\alpha, \beta, \gamma)$. The planes are defined by the vectors $\vec{P} = (l, m, n)$ and $\vec{n} = (\lambda, \mu, \nu)$, respectively.

Ray projection, from a source plane to a sensor plane, is modelled using the parametric equation of a 3D line (5.1). This allows each ray to be described in terms of a parameter t, which enables the calculation of the intersection points between the light rays and the sensor plane.

$$\frac{x-a}{\alpha} = \frac{y-b}{\beta} = \frac{z-c}{\gamma} (= t) \tag{5.1}$$

Where the intersection coordinates (x, y, z) occur within a target area, a hit occurs, representing illumination.

For any given combination of source plane, and sensor plane, the t parameter is calculated using the Line-Plane Intersection equation

$$t = \frac{\vec{n} \cdot \vec{P} - \vec{n} \cdot \vec{A}}{\vec{n} \cdot \vec{u}} \tag{5.2}$$

6 Results

6.1 Sensor Characterization

For the SensorCharacterization.tex file, you'd want to focus on the fundamental properties and performance of your photodiodes themselves, distinct from the other subsections. Here are some key elements that would belong specifically under SensorCharacterization:

Basic Photodiode Electrical Characteristics:

Dark current measurements Junction capacitance I-V characteristics in different lighting conditions Spectral response profiles (sensitivity vs. wavelength)

Individual Sensor Benchmarking:

Performance comparison between the 4 photodiodes (matching/differences) Responsivity measurements (A/W) Quantum efficiency calculations Detection threshold levels

Response Linearity:

Measurements showing linear range of the photodiodes Saturation point characterization Recovery time from saturation

Temperature Dependency:

Performance drift with temperature Baseline shift measurements Temperature compensation data

Aging/Stability Tests:

Long-term drift measurements Repeatability of measurements over time

This section should focus on the inherent properties of the photodiodes themselves - essentially providing the baseline characterization data that underpins all the other analysis. The other sections then build on this foundation by examining how these sensors perform when integrated into the complete system with amplification, angular positioning, enclosure effects, etc.

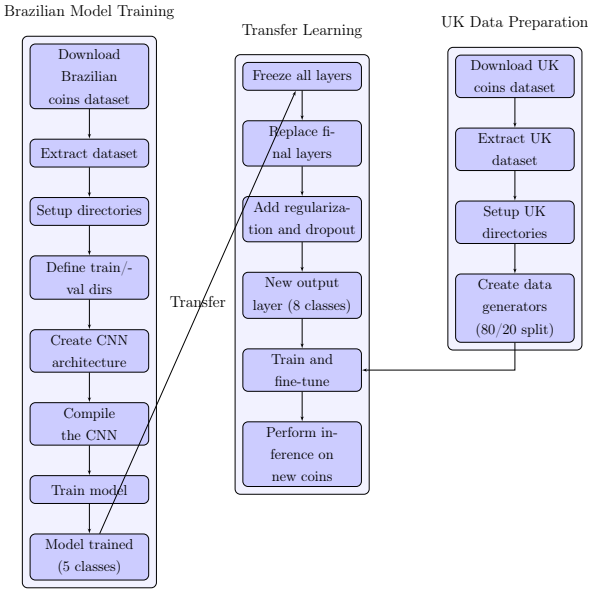

Figure 6.1: System Design Overview Flowchart

6.1.1 Functional Requirements

6.1.2 Design Approach

6.1.3 System Architecture

As shown in Figure 6.1 the system architecture consists of various components.

```
1 # Your code here
```

Listing 6.1: System Architecture Code Example

6.2 Amplification Performance

This section provides results of the amplifier performance.

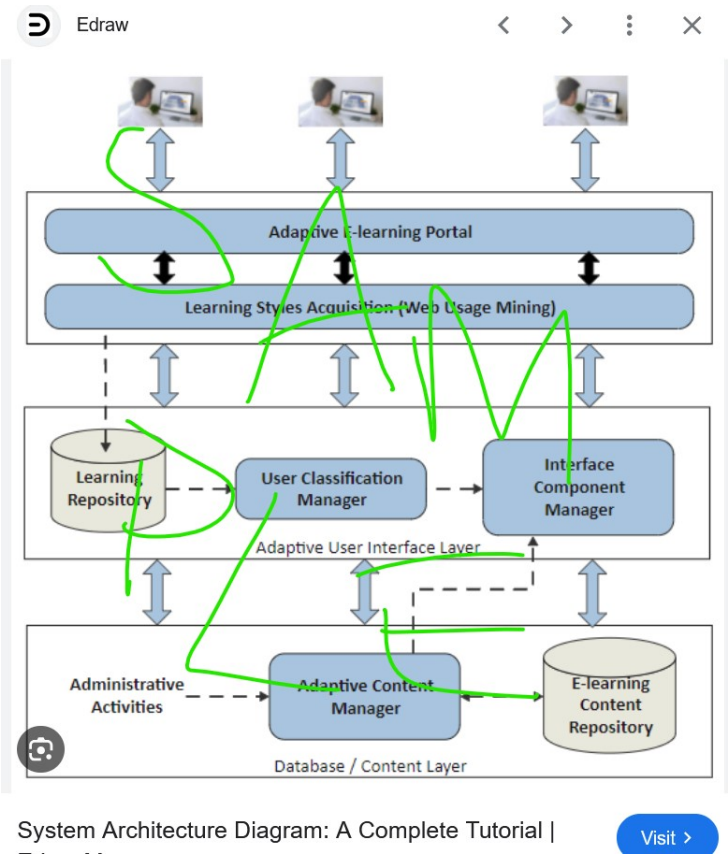

Figure 6.2: System Architecture Diagram

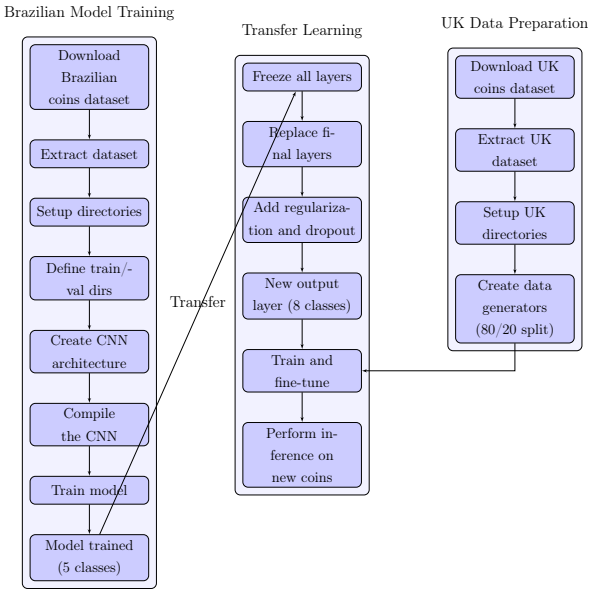

Figure 6.3: System Design Overview Flowchart

6.2.1 Functional Requirements

6.2.2 Design Approach

6.2.3 System Architecture

As shown in Figure 6.3 the system architecture consists of various components.

```
# Your code here
```

Listing 6.2: System Architecture Code Example

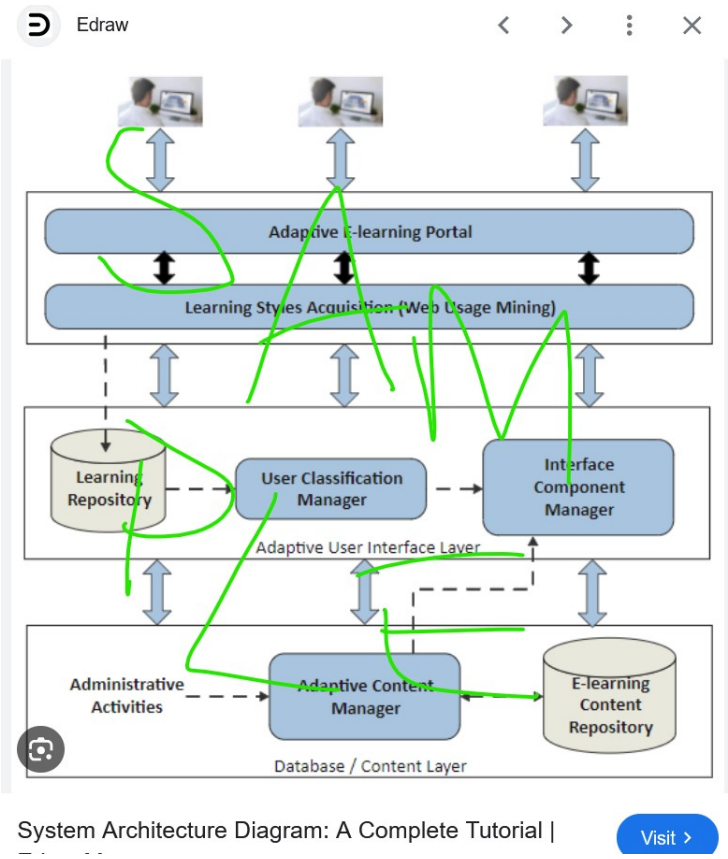

Figure 6.4: System Architecture Diagram

6.3 Photodiode Angular Response

This section discusses the results of the response of the solar sensor to angular changes of the light source.

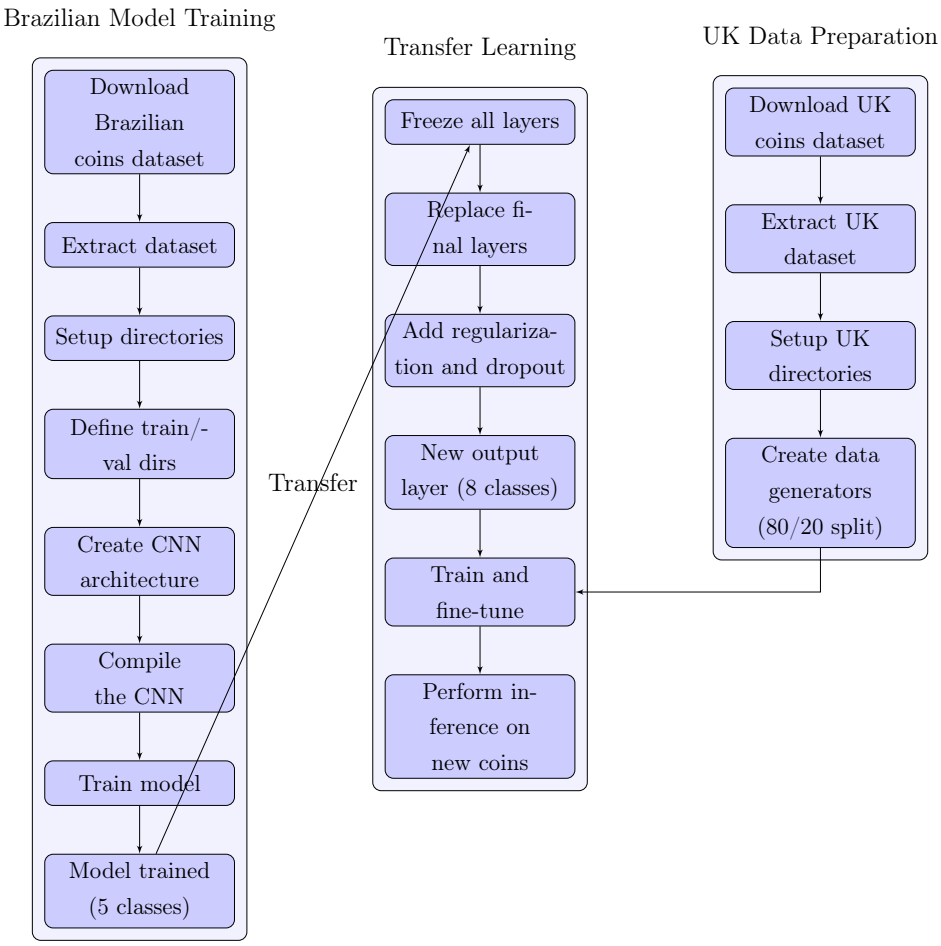

Figure 6.5: System Design Overview Flowchart

6.3.1 Functional Requirements

6.3.2 Design Approach

6.3.3 System Architecture

As shown in Figure 6.5 the system architecture consists of various components.

```
# Your code here
```

Listing 6.3: System Architecture Code Example

6.4 Enclosure Effectiveness

This section discusses the effectiveness of the Photodiode enlosure.

6.5 Data Acquisition System Evaluation

This section provides results related to the Arduino DAQ.

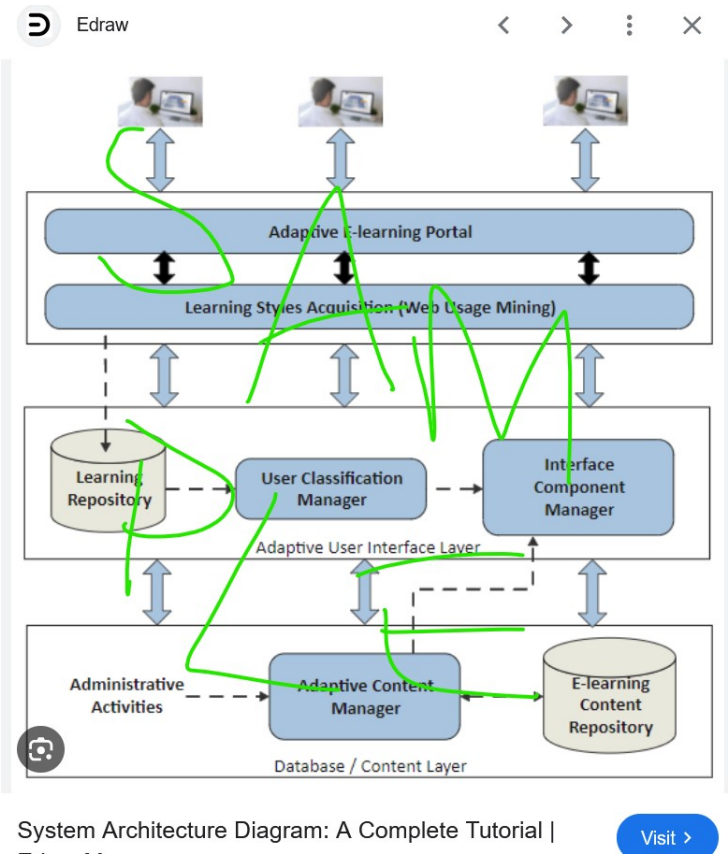

Figure 6.6: System Architecture Diagram

6.6 System Performance Analysis

6.6.1 Operational Constraints Identified

6.6.2 Environmental Factors Impact

```
// Environmental test results
// Temperature, ambient light, and vibration effects
```

Figure 6.7: Environmental Testing Results

6.6.3 System Stability and Repeatability

6.6.4 Recommendations for Improvement

6.7 Comparative Analysis

This section compares the simulation with the prototype results.

6.7.1 Breadboard vs. Stepboard Results

6.7.2 Iteration Improvements Analysis

6.7.3 Performance Against Design Requirements

The performance ...

6.7.4 Design Evolution Assessment

The what now?

6.8 System Limitations And Considerations

This section discusses the limitations and future work.

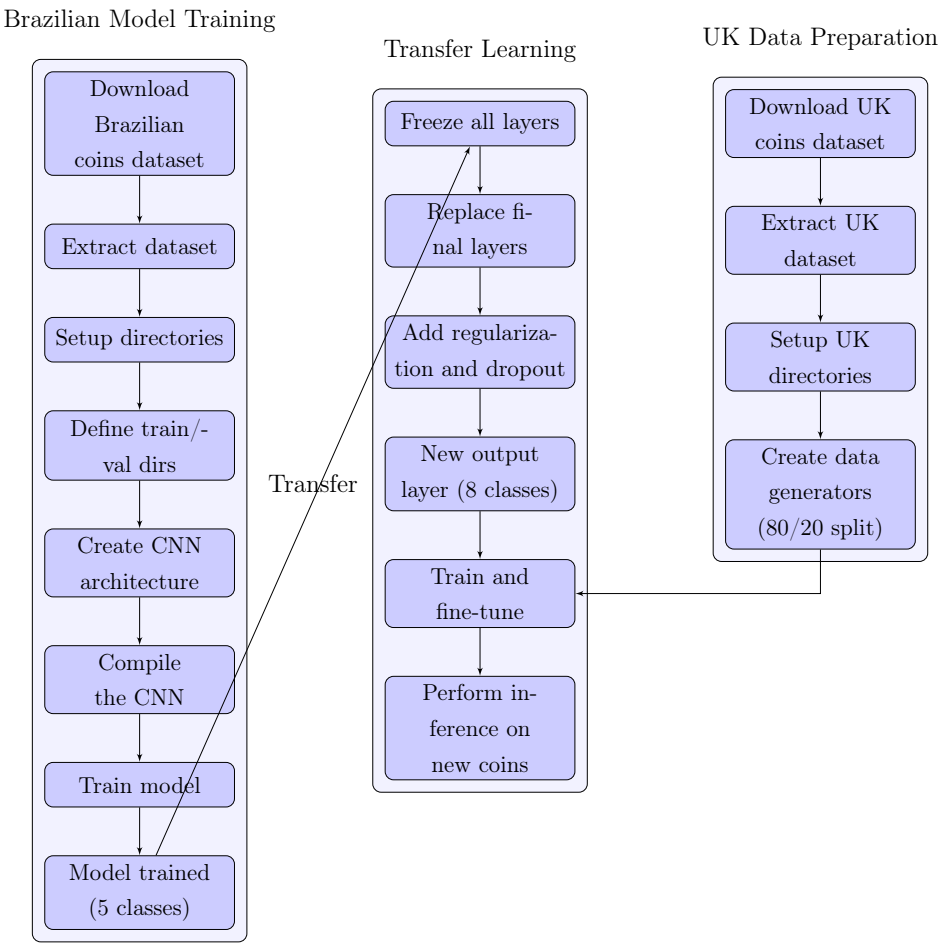

Figure 6.10: System Design Overview Flowchart

6.8.1 Functional Requirements

6.8.2 Design Approach

6.8.3 System Architecture

As shown in Figure 6.10 the system architecture consists of various components.

```
# Your code here
```

Listing 6.4: System Architecture Code Example

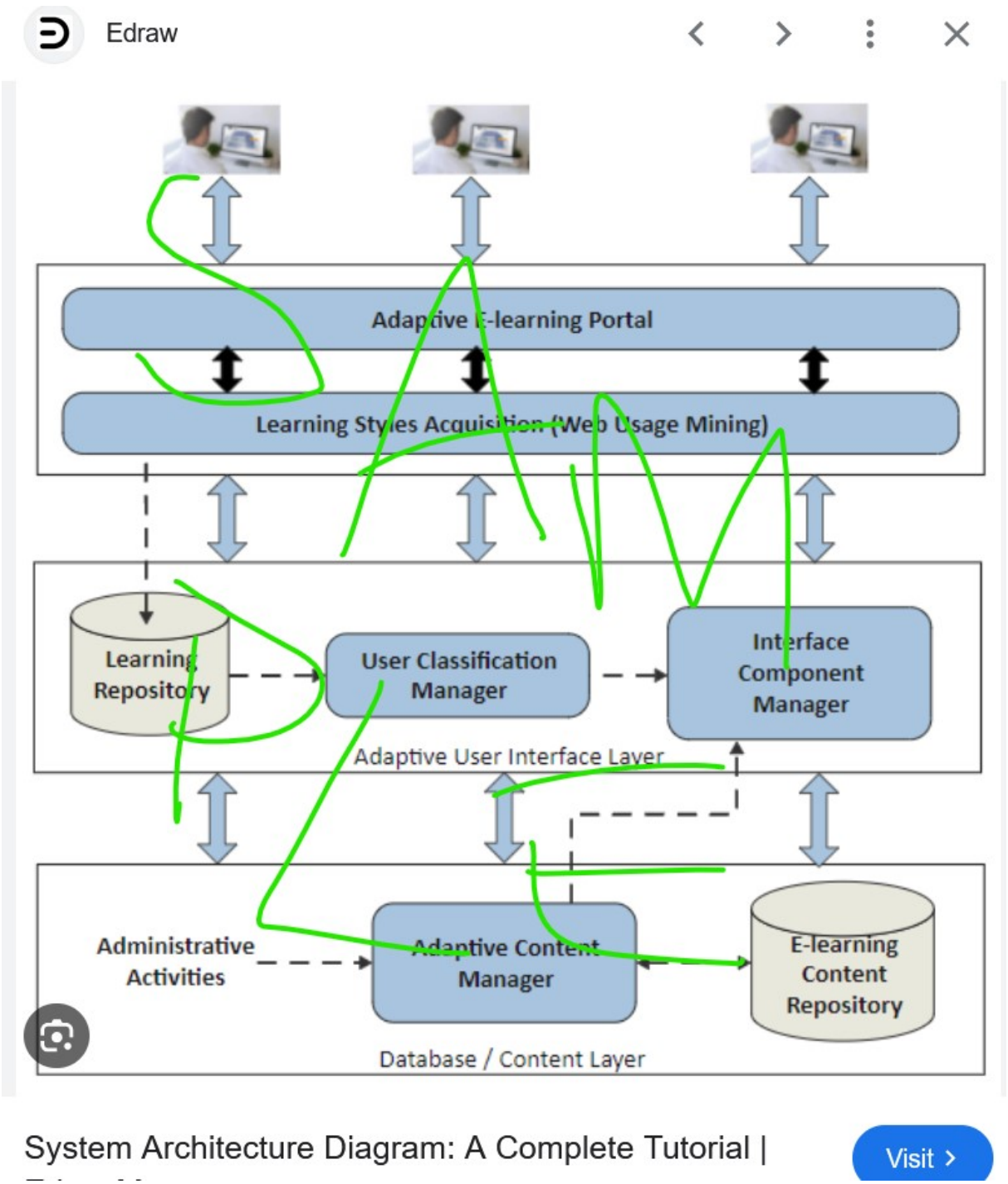

Figure 6.8: Overall System Performance Analysis

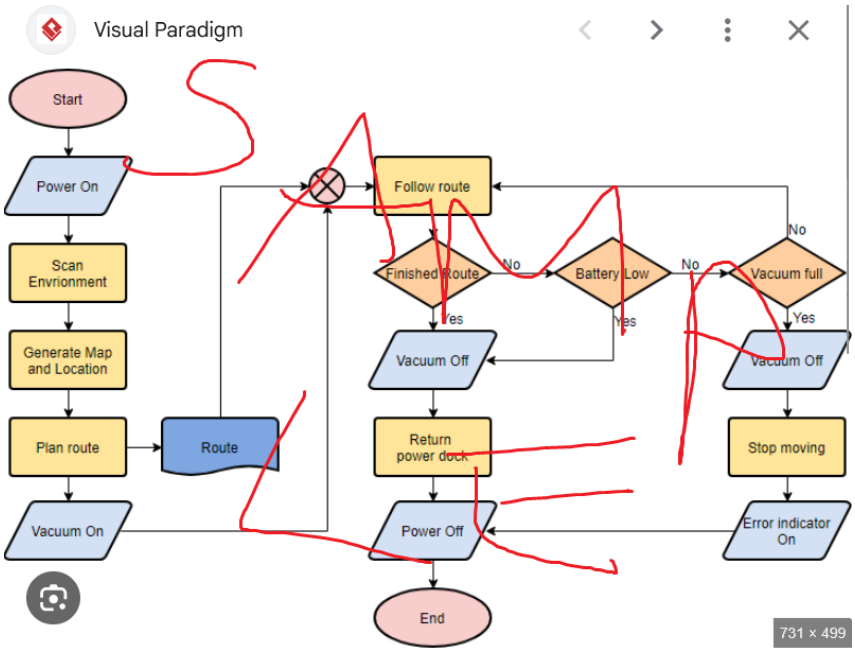

Figure 6.9: Prototype Iteration Comparison

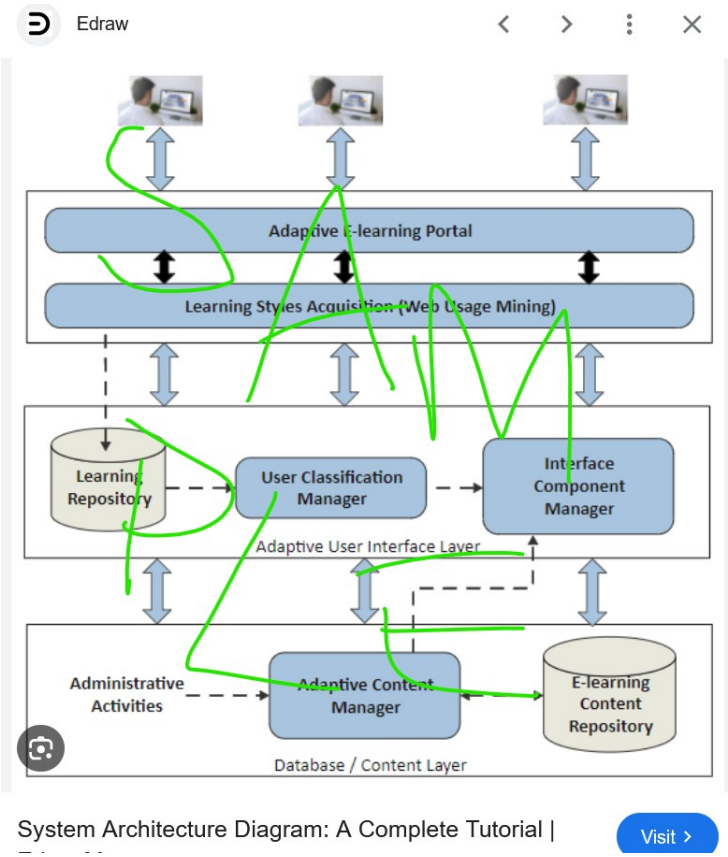

Figure 6.11: System Architecture Diagram

7 Conclusions

8 FutureWork

Bibliography

[1] J. Puig-Suari and C. Turner, "Development of the standard cubesat deployer and a cubesat class picosatellitel," 2001.

[2] K. S. Balaji, B. S. Anand, P. M. Reddy, V. C. B, M. D. P. Lingam, and V. K, "Studies on attitude determination and control system for 1u nanosatellite," *ICCPCT*, pp. 616–625, 2023.

[3] I. Lopez-Calle and A. I. Franco, "Comparison of cubesat and microsat catastrophic failures in function of radiation and debris impact risk," *Scientific Reports*, vol. 13, no. 1, -01-07 2023.

[4] R. W. Schafer, Oppenheim, and Schafer, *Discrete-Time Signals and Systems*. India: Pearson, 2013.

[5] Atmel, "Atmega328p 8-bit avr microcontroller with 32k bytes in-system programmable flash datasheet," 2015, 7810D-AVR-01/15. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[6] N. I. Shopov, S. Hannisse, and S. Gupta, "Renewable energy demonstrator (red)," Glasgow Caledonian University, Tech. Rep., 2022.

[7] G. L. G. Burbui, U. Reggiani, and L. Sandrolini, "Prediction of low-frequency electromagnetic interferences from smps," *ISEMC*, vol. 2, pp. 472–477, 2006.