

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

DESIGN AND IMPLEMENTATION OF PHOTOVOLTAIC ENERGY HARVESTING AUTOMATON

by
Iskandar Askarov

Global domestic electricity consumption has been rapidly increasing in the past three decades. In fact, from 1990 to 2020, consumption has more than doubled from 10,120 TWh to 23,177 TWh [1]. Moreover, consumers have been turning more towards clean, renewable energy sources such as Photovoltaic. According to International Energy Agency, global Solar power generation alone in 2019 has reached almost 3% [4] of the electricity supply. Even though the efficiency of photovoltaic panels has been growing, presently, the highest efficiency solar panels available to an average consumer range only from 20%-22% [14]. Many research papers have been published to increase the harvesting efficiency as well as the monitoring implementations for photovoltaic.

In this thesis, we would like to introduce our design of a photovoltaic energy harvesting automaton – “IziBu”. We have aimed our design objectives to be autonomy and scalability. Our design includes dual-axis Sun tracking and sensory data acquisition and processing. We have made our system to be powered solely by an attached photovoltaic panel using an uninterruptable power supply (UPS) and an external battery. Additionally, in case of a complete loss of a charge in the battery, our system has an ability to wake up when battery charge reaches the certain level. The automaton can operate as a standalone or a cluster member. Entire automaton is controlled by the scheduler we have designed.

**DESIGN AND IMPLEMENTATION OF PHOTOVOLTAIC ENERGY HARVESTING
AUTOMATON**

by
Iskandar Askarov

**A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
Department of Computer Science**

December 2021

Blank Page

APPROVAL PAGE

**DESIGN AND IMPLEMENTATION OF PHOTOVOLTAIC ENERGY HARVESTING
AUTOMATON**

Iskandar Askarov

Dr. Jing Li, Thesis Advisor Date
Assistant Professor of Computer Science, NJIT

Dr. Shantanu Sharma, Committee Member Date
Assistant Professor of Computer Science, NJIT

Dr. Xiaoning Ding, Committee Member Date
Associate Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Iskandar Askarov

Degree: Master of Science

Date: December 2021

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
New Jersey Institute of Technology, Newark NJ, 2021
- Bachelor of Science in Electrical and Computer Engineering,
New York Institute of Technology, New York NY, 2012

Dedication

I want to dedicate this thesis to my Father Khusnitdin and my Mother Saida. Thank you for everything you have done for your children. I owe you everything in my life.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis Advisor Dr. Jing Li for guidance and mentorship. For always keeping the open-ended communication and helping without constraints of time. I am inspired by her humbleness, achievements, and sincerity towards mentoring throughout the research period.

Additionally, I would like to thank Committee Members Dr. Shantanu Sharma and Dr. Xiaoning Ding for giving me honest feedback.

I want to thank my Father for his guidance and advisement on the technical aspects of this paper. This thesis would not have happened without you, Dad.

I want to thank my beautiful wife Nigina for always being there for me and supporting me through tough times.

I want to thank Matthew Siliato, for his support and coverage throughout the research period.

CONTENT

Chapter	Page
1 INTRODUCTION.....	1
2 IMPLEMENTATION.....	3
2.1 Hardware.....	3
2.1.1 Schematic.....	3
2.1.2 Photovoltaic Panel.....	4
2.1.3 Metal Structure.....	5
2.1.4 Single Board Computer.....	5
2.1.5 UPS with attached Li-Po battery.....	6
2.1.6 Current Sensor.....	7
2.1.7 Relay Module.....	7
2.1.8 LDR.....	8
2.1.9 Servo motors.....	9
2.2 Software.....	10
2.2.1 Setup and Calibration.....	11
2.2.2 IPC Mechanism.....	11
2.2.3 LDR Sensory Controller.....	12
2.2.4 Servo Motor Controller.....	14
2.2.5 Battery Controller.....	14
2.2.6 Master Controller.....	14
2.2.7 Flowchart.....	17
3 EXPERIMENTAL RESULTS.....	19
3.1 Initialization and Setup.....	19

3.2 Current and LDR Sensory Results..... 19

3.3 Li-Po Battery Management..... 20

3.4 Autonomy..... 21

4 CONCLUSION..... 22

LIST OF TABLES

Table		Page
2.1	Characteristics Of Servo Motors.....	13
3.1	Description Of Run Levels.....	21

LIST OF FIGURES

Figure		Page
2.1	Schematic of hardware components.....	5
2.2	Connection diagram of photovoltaic cells.....	6
2.3	Previous implementation of metal structure.....	7
2.4	Current implementation of metal structure.....	7
2.5	Raspberry Pi Zero W.....	8
2.6	PiJuice Zero.....	9
2.7	Connection diagram of the ACS712(30A).....	10
2.8	Connection diagram of the PCF8591.....	10
2.9	Connection diagram of the relay module.....	11
2.10	3D Model of LDR Housing.....	12
2.11	Connection diagram of LDRs.....	12
2.12	Mounting points of LDRs.....	13
2.13	Structure of Motor table in SQLite database.....	16
2.14	Cluster communication mode of IziBu.....	16
2.15	LDR marked locations on photovoltaic panel.....	17
2.16	Implementation of Analog-Digital conversion.....	18
2.17	Pseudo code of the DAST algorithm.....	19
2.18	Flowchart of the DAST algorithm.....	19
2.19	Flow of the Servo Motor Controller thread.....	20
2.20	First part of flowchart.....	24
2.21	Second part of flow chart.....	25
3.1	Screenshot of initialization.....	26

3.2 Screenshot of Current and LDR Sensory results..... 27

3.3 DC power supply..... 27

3.4 Screenshot of Li-Po battery management component..... 28

3.5 Battery discharge graph..... 29

LIST OF EQUATIONS

Equation		Page
2.1	Ohm's Law.....	6
2.2	Equation of parallel photovoltaic cells.....	6
2.3	Voltage divider equation for LDRs.....	12
2.4	PWM frequency equation.....	14
2.5	Equation to convert LDR's analog output to digital.....	18
2.6	Angular position equation for base servo.....	18
2.7	Angular position equation for top servo.....	18
2.8	Sleep timer equation.....	18
2.9	Equation to calculate the duty cycle of the servo.....	20
2.10	Arithmetic average of the angular position of last three days.....	22
2.11	Equation to convert ACS712 analog output to digital.....	22

LIST OF SYMBOLS

SBC	Single Board Computer
TWh	Terawatt-Hour
PV	Photovoltaic
Li-Po	Lithium Polymer
SCADA	Supervisory Control and Data Acquisition
DAST	Dual Axis Solar Tracking
SAST	Single Axis Solar Tracking
AD	Analog-Digital
LDR	Light Dependent Resistor
IPC	Inter-Process Communication
STC	Standard Test Conditions
PWM	Pulse-Width Modulation
UPS	Uninterruptable Power Supply

CHAPTER 1

INTRODUCTION

Photovoltaic also known as a solar is a widely used renewable energy source around the world. According to International Energy Agency, global Solar power generation in 2019 has reached almost 3% [4] of the electricity supply. In comparison to wind energy, photovoltaic solutions are noise-free and require minimal maintenance. However, one of the drawbacks of photovoltaic is its low efficiency. At the time of this writing, an average consumer has access to only 20%-22% [14] efficient photovoltaic panels. Additionally, the highest productivity window is even more limited on fixed-tilted photovoltaic installation.

Many research papers have been published to increase the efficiency of the photovoltaic by introducing single-axis and dual-axis solar tracking implementations. For instance, to maximize the power transfer from the photovoltaic panel, Aboubakr El Hammoumi et al. [9] implement DAST system with a voltage divider using four light-dependent resistors (LDR) mounted on four corners of the photovoltaic panel. Output of the voltage divider is fed into analog pins of Arduino Uno with ATmega328 microcontroller. Their implementation has resulted in a 36.25% increase in power transfer. In addition to solar tracking solutions, data monitoring and acquisition have been widely researched and implemented within the industry and academia. A paper by Lawrence O. Aghenta et al. [13] implements a very interesting SCADA design. A voltage and current sensors are connected to the ESP32 board, which in turn is fed into the IoT device. However, their implementation requires an external power source for an IoT device to operate. Another SCADA proposal by Tariq Iqbal et al. [10] implements a similar approach of monitoring energy supply by photovoltaic and claims that a battery bank also powers an energy management system. However, the author does not give any information on how ESP32 is powered through a battery bank as no schematic is provided. Hence, we assume

that an external power source still powers ESP32. A paper by Layse Nascimento et al. [12] implements an off-grid, portable SCADA solution on Arduino Mega2560. Where they successfully implemented autonomy using an 8Ah battery. Given that it is a home automation system, their design does not have a DAST implementation.

In this thesis, we would like to introduce our design of a photovoltaic energy harvesting automaton – “IziBu”. We have studied a number of research papers in supervisory control and data acquisitions (SCADA) and dual-axis solar tracking (DAST) methods and implementations. Therefore, we have aimed our design objectives to be autonomy, scalability. In addition, we have also tried to improve on some of the methods and implementations. We have made our system to power solely by an attached photovoltaic panel for autonomy. Our automaton is powered using an uninterruptable power supply (UPS) and an external Li-Po battery. Additionally, in case of a complete loss of a charge in the battery, our system has an ability to wake up when battery charge reaches a certain level. This logic has been implemented in the scheduler of the IziBu.

We have designed our automation to operate as a standalone or a cluster member by implementing a leader-follower model. Our system consists of electronic components such as a current sensor for monitoring, photoresistors and servo motors for DAST, analog-digital converter, relay module, UPS and Li-Po battery. Automation is operated by a Raspberry Pi – a single-board computer (SBC). Additionally, we have designed and implemented an event-driven scheduler. The majority of the code has been written in C, with some implementations in Python programming languages. The main duties of the scheduler are

- Control DAST.
- In cluster mode, send or receive and process angular positions for servo motors.

- Guaranteeing that photovoltaic panels face the Sun at sunrise by recording historical positions of servo motors.
- Real-time monitoring of an electric current supplied by photovoltaic panel.
- Track the charge levels of the Li-Po battery and charge whenever necessary.
- Control the electric current flow from the photovoltaic panel.

The scheduler divides the system into four run levels: GREEN, YELLOW, RED, and BLACK. Every run level indicates battery charge, and scheduler's behavior depends on the system's run level.

CHAPTER 2

IMPLEMENTATION

On the high level, the implementation of IziBu consist of multiple parts – electronic components used to build the setup and the scheduler that was written to control the automaton. In this section, we will describe hardware or electronic components that have been used to build the automaton.

2.1 Hardware

The hardware part of the automaton consists of the metallic structure to mount the photovoltaic panel. Photovoltaic cells with parallel connection to increase electric current output. Current sensor – ACS712(30A), connected to the photovoltaic panel on one end and PCF8591 AD converter on the other. Uninterruptable Power Supply – PiJuice mounted on top of the Raspberry Pi Zero W with attached 500mAh Li-Po battery. Three light-dependent resistors (LDR) with voltage divider circuit for DAST algorithm implementation. Normally closed relay module to control the current from photovoltaic panels.

2.1.1 Schematic

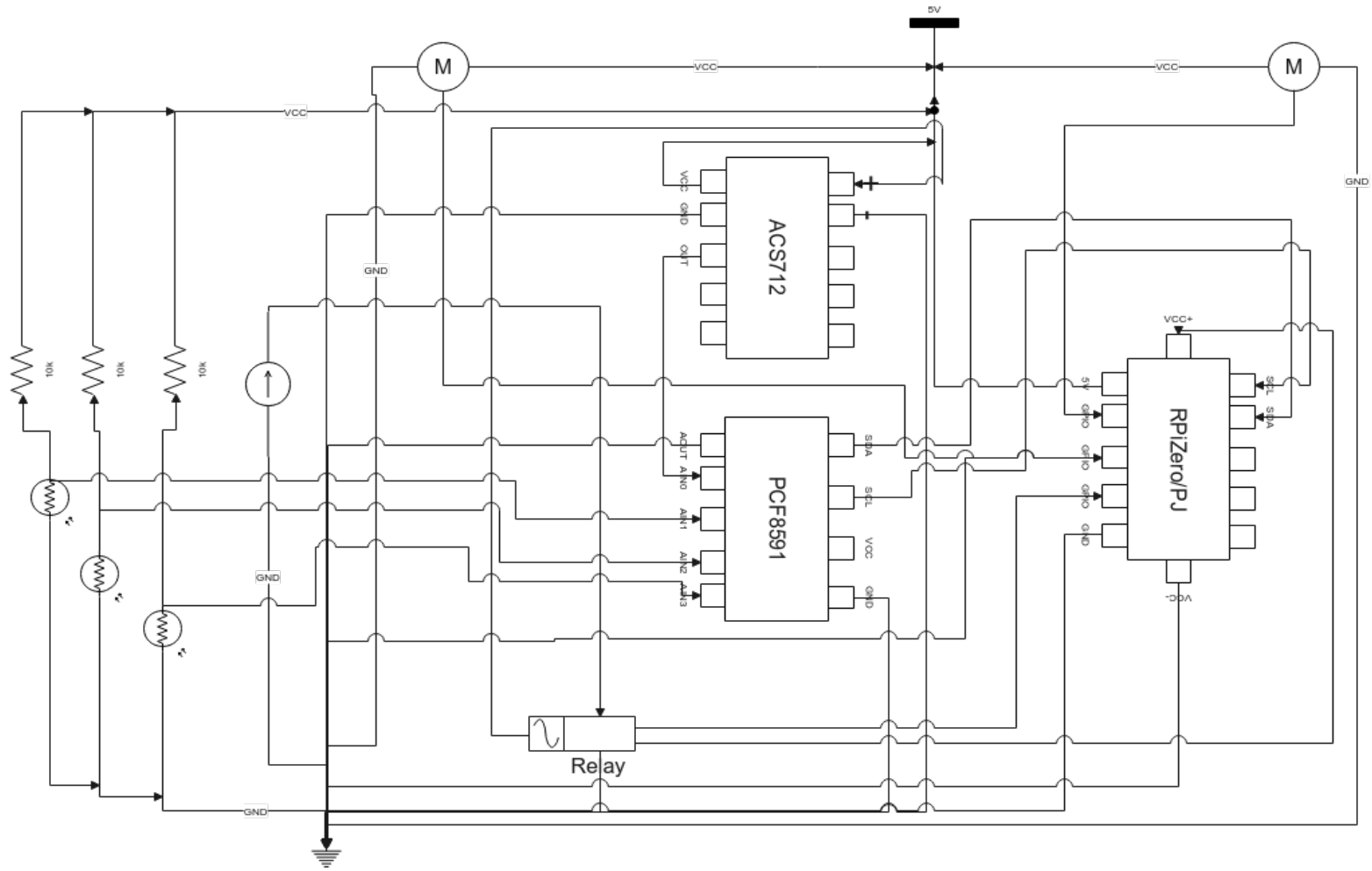


Figure 2.1 Schematic of hardware components.

2.1.2 Photovoltaic Panel

Photovoltaic also known as a Solar is a technology that converts sunlight into a direct electrical current. The most common type of photovoltaic panels is silicon-based. Traditionally, Solar panels consist of smaller photovoltaic cells connected in series and/or parallel to increase the output voltage and/or current. However, given that our objective is to measure the output current from the photovoltaic panel. We have connected our solar cells in parallel. According to a manufacturer, our photovoltaic cells are rated 2.5Watts and 5V and have a 17% efficiency. Thus by equation 2.1,

$$I = \frac{P}{V} \quad (2.1)$$

Where $P = 2.5W$ and $V = 5V$ yields $0.5A$ per cell. Hence, in theory, a parallel connection from six photovoltaic cells should generate up to $3A$ of current under STC[8]:

$$I = \sum_1^6 i = (i_1 + i_2 + i_3 + i_4 + i_5 + i_6) \quad (2.2)$$

The figure below demonstrates the parallel connection of the photovoltaic panels.

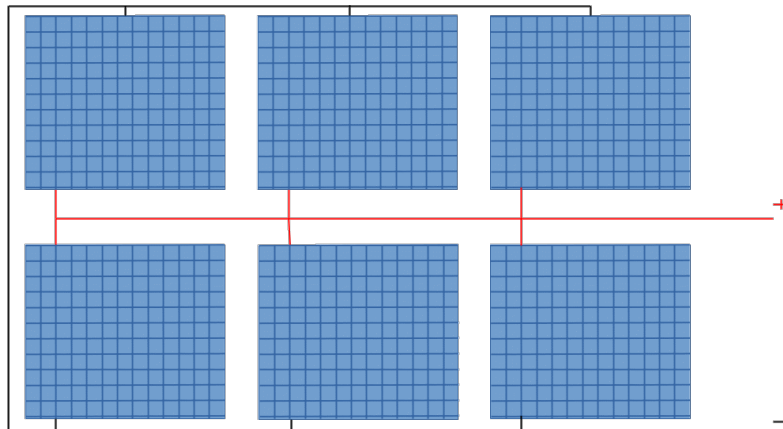


Figure 2.2 Connection diagram of photovoltaic cells.

2.1.3 Metal Structure

The criteria for the design of the metal structure is to be able to mount all components required for the automaton and allows to rotate photovoltaic panels with two degrees of freedom. These includes photovoltaic panel, servo motors, a single board computer, LDRs, current sensor, analog-digital converter and any circuits that was built. Thus, while working on this project, we have tried and built various structures. One of the designs we have tried to implement is listed in figure 2.3. Two single channel sliders mounted on the plank with photovoltaic panels attached between them. Downside of this implementation was the weight of the structure as the base servo we used to rotate the structure did not have enough torque to rotate and hold the firm position. Eventually, we decided to repurpose the base frame from the robotic arm kit we had available. The structure allows us to two mount (base, top) servos for the two degrees of freedom, photovoltaic panel, a single board computer, and other components. Figures 2.4 demonstrations the current metal structure design we have implemented.

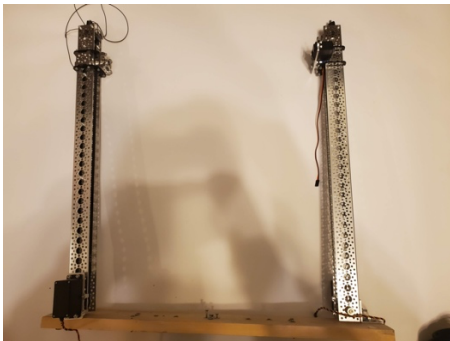


Figure 2.3 Previous implementation.

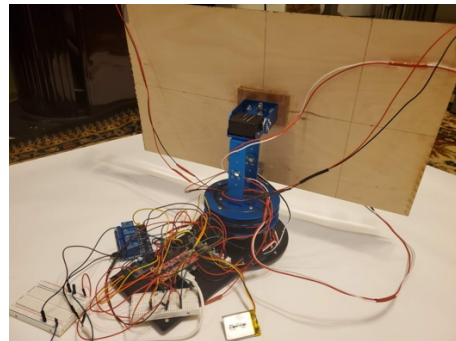


Figure 2.4 Current implementation.

2.1.4 Single Board Computer

One of the main ideas behind the design of the IziBu is autonomy and scalability. As we stated previously, a system can be operated as a standalone node or a cluster member. Thus, we decided

to implement our solution on a Raspberry Pi Zero W (Rpi) [6], which is the backbone of the automaton. Employing Rpi allows us to use Linux operating system with all underlying mechanisms to build our software component. Another benefit is that Rpi Zero W is known for its low power consumption [16]. Specifications for our model are listed below [3]:

- Dimensions: 65mm × 30mm × 5mm
- SoC: Broadcom BCM2835
- CPU: ARM11 running at 1GHz
- RAM: 512MB
- Wireless: 2.4GHz 802.11n wireless LAN
- Bluetooth: Bluetooth Classic 4.1 and Bluetooth LE
- Power: 5V, supplied via micro USB connector
- Video & Audio: 1080P HD video & stereo audio via mini-HDMI connector
- Storage: MicroSD card
- Output: Micro USB
- GPIO: 40-pin GPIO, unpopulated
- Pins: Run mode, unpopulated; RCA composite, unpopulated
- Camera Serial Interface (CSI)

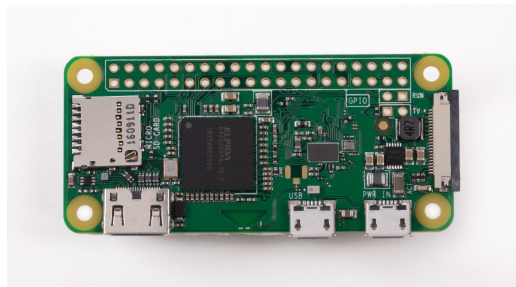


Figure 2.5 Raspberry Pi Zero W.

Since our design has no use for many ports such as HDMI, Bluetooth, and onboard LED, we have decided to power them down, which allows us to save more power on the attached Li-Po battery.

2.1.5 UPS With Attached Li-Po Battery

As we described previously, we have tried to design IziBu with autonomy in mind - to be able to run the system off-grid. Thus, we used the PiJuice Zero power supply, which perfectly integrates with our model of SBC. Additionally, we have attached a 500mAh Li-Po battery to UPS with a built-in charge controller. That said, our SBC is powered by Solar panels that charge the attached Li-Po battery during the daytime, and the charge controller protects the battery from overcharging. When the Sun is below the horizon, the Li-Po battery is used as a power source for an automaton.

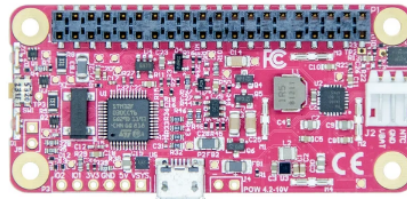


Figure 2.6 PiJuice Zero.

Additionally, PiJuice Zero allows us to get the real-time status of the attached Li-Po battery using several libraries available on the internet [5], which we have integrated with our software component.

2.1.6 Current Sensor

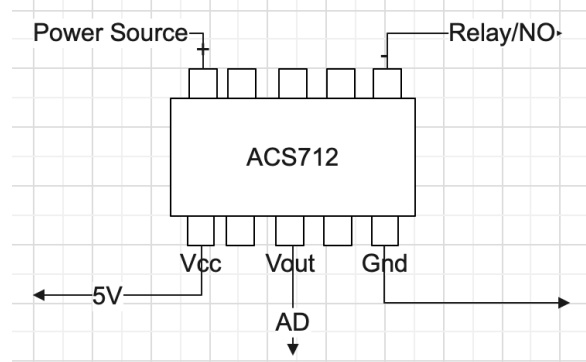


Figure 2.7 Connection diagram of the ACS712(30A).

To measure the current from photovoltaic panels, we have used a low-cost ACS712 (30A). The figure 2.7 demonstrates the connection diagram of the component. ACS712 is based on the Hall-Effect principle [2] that measures an electric current flow and outputs the analog signal. The component operates on 5V and outputs 66mV/A. This means for every Ampere increase, Voltage from ACS712 is increased by 66mV. Given that, Rpi does not have analog ports available. We needed to convert the output signal from ACS712 to digital and used a PCF8591 8-bit AD converter. AD converter was connected to serial ports SCA and SDL on Rpi. 8-bit conversion will yield a range from 0-255. We will describe the conversion method in the Master controller part of the scheduler section of Chapter 2. Figure 2.8 demonstrates connection diagram of PCF8591 AD converter.

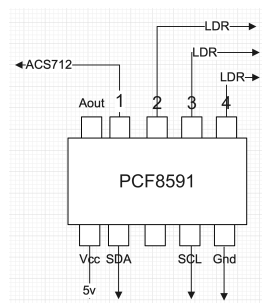
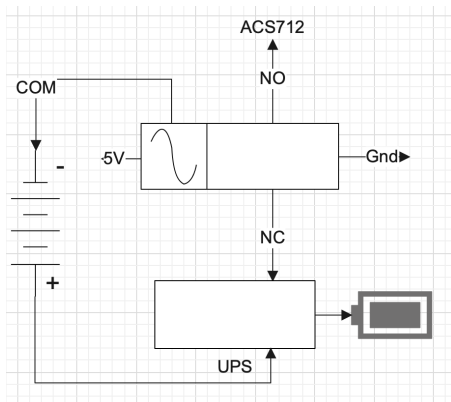


Figure 2.8 Connection diagram of PCF8591 AD converter.

2.1.7 Relay Module

We employed a normally closed relay module to control the flow of the current. Rpi powers the relay module. Relay is also connected to one of the GPIO pins on Rpi. Given that Rpi controls the entire system charging of Li-Po battery takes precedence. Hence, we have connected the NC terminal of the relay to the UPS and NO terminal to the ACS712 current sensor. This allows us to charge Rpi when there is a flow of a current supply from the Solar panels, even when the onboard Li-Po battery is drained. When the battery charge reaches a certain level, the relay is switched to redirect the current to the ACS712 sensor.



Normally closed terminal: UPS

Normally open terminal: ACS712(30A)

Common terminal: (-)

Figure 2.9 Diagram of the relay module.

2.1.8 Light Dependent Resistors

LDRs are a component that is widely used to detect the intensity of light. Resistance of LDR decreases with the increasing intensity of light, resulting in a drop of the voltage across the resistor.

We have used LDR as the main component when implementing active-sun tracking in our design. LDR was mounted in a 3D printed housing in Figure 2.5.

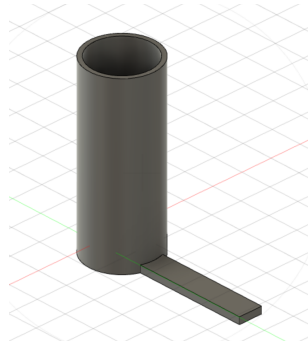


Figure 2.10 3D Model of LDR Housing.

We have taken the similar approach described in the paper by El. Hammoumi et al. [9], and implemented a voltage divider circuit. El. Hammoumi et al. [9] use four LDRs placed in four corners of the photovoltaic panel. Our algorithm uses only three LDRs placed in two top corners and one bottom-center of the photovoltaic. Figure 2.6 shows the position of the mounted LDR on the photovoltaic panel.

We measure the intensity of light by measuring the output voltage from the voltage divider as shown in figure 2.11 and Equation 2.3.

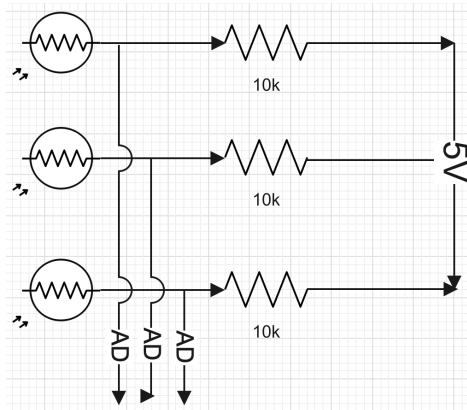


Figure 2.11 Connection diagram of LDRs.

$$V_{AD} = V_{in} \frac{R_{LDR}}{R + R_{LDR}} \quad (2.3)$$

The value R_{LDR} varies by the light intensity thus the output voltage (V_{AD}) in equation 2.3 produces an analog signal reaching 5 volts max at high light intensity. In order to quantify the output, we needed to convert it to a digital signal. Hence, every LDR was connected to a PCF8591 AD converter as shown in figures 2.8 and 2.11 . We describe the method of conversion in the Sensory component of the scheduler.

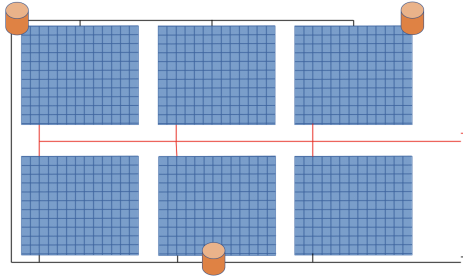


Figure 2.12 Mounting points of LDRs.

2.1.9 Servo Motors

In order to get the most optimal power transfer from photovoltaic panels, we have implemented sun-tracking with two degrees of freedom. Two DC servo motors were used to implement this functionality. Some of the technical characteristics of servo motors are listed in Table 2.1

Table 2.1 Characteristics of Servo Motors

Servo model	LD-1501MG	LDX-218
Mounting point	Base	Top
Pulse-Width	500 ~ 2500 μ sec (0 ~ 180deg)	500~2500us to (0~180deg)
Operating voltage	6-7.4V	6-8.4V
Stall torque	13kg.cm @6V; 15kg.cm @6.5V; 17kg.cm @7V	15 kg·cm @ 6V 17 kg·cm @ 7.4V
Operating speed	0.16s/60° @ 7.4V	0.16 sec/60° @ 7.4V
Running current	100mA	100mA
Weight	60g	56g
Dimensions	40*20*40.5mm	40*20*51.4mm
Deadband	11.1 μ sec	11.1 μ sec

One of the features that distinguish servos from other motors is the ability to provide position feedback. Unfortunately, given that the servos we have employed were primarily intended for hobbyists, they lacked feedback. Hence, we have used a file-based SQLite database to calibrate and record the historical position of our servos.

Since using a servo controller module requires an additional power, we have opted to control servos directly from RPi using hardware PWM instead. The base frequency of PWM chip on Rpi is 19.2Mhz. Thus, in order to get PWM frequency of 50Hz, which is equivalent to 20ms, we set the clock divider to 192 and PWM range to 2000. As shown in equation 2.3 where f_{base} base frequency of PWM, n – clock divider and t – PWM range.

$$f_{out} = \frac{f_{base}}{n * t} = \frac{19.2 * 10^6 Hz}{192 * 2000} = 50Hz \quad (2.4)$$

Implementing servo control using a hardware PWM 50Hz frequency allowed to precisely control our rotational angles as well as the speed of the servo. Prior to implementing hardware PWM, we have also used software pulse-width modulation. That resulted in inaccurate rotations, 7-8 degree jumps, and constant jitters. We have ended up damaging multiple servos while trying to implement software PWM. Upon our investigation, we have discovered that software implementation of pulse-width modulation uses CPU cycles for pulse generation. That said, the thread that generates pulse-width relies on process scheduler of the Linux - Completely Fair Scheduler (CFS). Thus, it is up to CFS to decide when to execute the process for pulse generation.

2.2 Software

The second part of the IziBu is the scheduler. In order to avoid a CPU overhead of interpreter languages such as garbage collection and bytecode translations, we have specifically chosen the C to be the main programming language for implementation. That said, the scheduler has been written as a residential multi-threaded program allowing us to implement IPC using sockets and named pipes easily. We also have set up the scheduler as a system-level process with the highest priority in Linux. Thus, upon system restart, the IziBu scheduler is started automatically. The scheduler itself consists of multiple subcomponents such as Setup and Calibration, IPC mechanism, LDR Sensory, Servo motor control and reactive battery charge controller [15], and Master Controller. We will describe each of them in this chapter.

2.2.1 Setup and Calibration

Setup and Calibration require manual intervention by a user. As we have stated previously, we have used multiple servos to achieve the current state of the IziBu. Since we do not require our servos to have a full rotation of 0-180* due to obstacles of the metal structure, we use this setup to set the limit of the servos as well as get the mapping values for PWM to angular values of the servo. Calibrated data is stored in a file-based SQLite database. During the initial Setup phase system generates the database and allows the user to indicate the direction of the servo, whether it rotates up/down or left/right. These values are useful for a user during the actual operation of the automaton, as every operation is recorded in the log file. Additionally, the user is able to run the servos and set the hard stop for each servo. All these values are recorded in the Motor table listed below.

```

CREATE TABLE Motor
(
    HostNameVARCHAR(6) NOT NULL DEFAULT('MASTER'),
    DirectionIDSMALLINT NOT NULL DEFAULT(0)
    MotorIDSMALLINT NOT NULL DEFAULT(0)
    RealRotationalDegreeSMALLINT NOT NULL DEFAULT(90),
    HardwareRealLowRangeValueSMALLINTNOT NULL DEFAULT(0),
    HardwareRealHighRangeValueSMALLINTNOT NULL DEFAULT(0),
    SoftwareRealLowRangeValueSMALLINTNOT NULL DEFAULT(0),
    SoftwareRealHighRangeValueSMALLINTNOT NULL DEFAULT(0)
);

```

Figure 2.13 Structure of Motor table in SQLite database.

2.2.2 IPC Mechanism

As we have stated previously, we designed IziBu to operate in standalone as well as a member of a cluster. For a clustering mode, we have implemented TCP/IP sockets in a second iteration of the scheduler. In clustering mode, follower nodes are fully operated by a leader node. Execution payloads are sent by a leader node to all follower nodes via TCP protocol. Thus, no calculation are done by the follower nodes. For clustering we require all nodes to be in the same WiFi network.

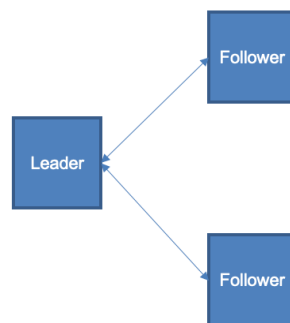


Figure 2.14 Cluster communication mode of the IziBu.

Given the complexity of the TCP protocol, an improvement to the current design can be achieved by employing UDP protocol. Simplicity of the UDP protocol allows the transmission of the

payload to the followers without an unnecessary overhead of the TCP such as synchronization, acknowledgement and lost packet retransmission. However, in case of the UDP it is best implement a heartbeat mechanism between leader and followers. This would allow for a leader node to know how many followers are in a ready/live state to receive the payload.

In standalone mode, we implemented named pipes. Each servo motor controller has a dedicated named pipe from which it receives a payload. The payload contains an angular degree for a servo to rotate. Payloads are sent by the Master controller calculated by DAST algorithm. We will describe the LDR sensory module and DAST algorithm in the next section.

2.2.3 LDR Sensory Controller

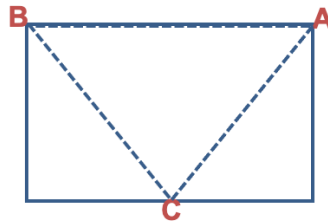


Figure 2.15 LDR marked locations on photovoltaic panel.

Main functionality of the LDR sensory module is to read, interpret the signals received by LDRs and run them through DAST algorithm. Due to trial-and-error, we have discovered that the most optimal performance of the DAST algorithm is achieved by creating an equilateral triangle.

Thus, as shown in the figure 2.15, we have mounted three LDRs on the top corners and bottom center of the photovoltaic panel. We have stated previously in the LDR section that we measure the output of the voltage divider circuit. As solar intensity increases LDR value increases as well. Which in turn outputs higher voltage in analog format. In order to interpret the analog value to

digital, we connect all outputs to an 8-bit PCF8591 AD converter and use the conversion formula below.

$$V = \frac{\text{AnalogInput} * 5.0 \text{ volts}}{255} \quad (2.5)$$

```
float digitalGetVoltagePCF8591(int aChannel, int aFileDescriptor)
{
    if (aFileDescriptor <= 0 || aChannel < 0) return 0.0;
    return analogGetValuePCF8591(aChannel, aFileDescriptor) * PCF8591_Voltage / 255.0;
}
```

Figure 2.16 Implementation of Analog-Digital conversion.

Function digitalGetVoltagePCF8591 is the implementation of the conversation formula.

AnalogOutput is returned by analogGetValuePCF8591 function. aChannel parameter is the channel we are reading. Thus, LDR sensory algorithm used for DAST is given in equations 2.5 - 2.7 and figure 2.17. Where variables A, B, and C are the LDR values, α_i is angular position for base servo, β_i is angular position for top servo and t is sleep time.

$$\alpha_i = \begin{cases} \alpha_{i-1} + 1 & \text{if } A > B \\ \alpha_{i-1} - 1 & \text{if } A < B \end{cases} \quad (2.6)$$

$$\beta_i = \begin{cases} \beta_{i-1} + 1 & \text{if } A > C \mid B > C \\ \beta_{i-1} - 1 & \text{if } A < C \mid B < C \end{cases} \quad (2.7)$$

$$t = \begin{cases} 10 \text{ sec} & \text{if } A = B \text{ and } A = C \text{ and } A \neq 0 \\ > 10 \text{ sec} & \text{if } A = B = C = 0 \end{cases} \quad (2.8)$$

Figure 2.15 is the pseudo code implementation of above equations.

```

While true
  If A = B = C = 0 then
    sleep = getSleepTime_BasedOnTimeOfDay()
  if A = B = C != 0 then
    sleep 10 seconds
  If A > B then
    increment angular value of base servo by 1
  If A < B then
    decrement angular value of base servo by 1
  If A > C or B > C then
    increment angular value of top servo by 1
  If B < C or B < C then
    decrement angular value of top servo by 1

```

Figure 2.17 Pseudo code of DAST algorithm.

Function `getSleepTime_BasedOnTimeOfDay` is just a shortened version of the logic that calculates sleep value. For example, at night time, there is no need to read LDR values. Additionally, sleep period helps with noise avoidance. Sleep times can be manually set. Once angular values are adjusted, they are pushed to a named pipe of Servo Motor controller. Figure 2.18 demonstrates flowchart of the DAST algorithm.

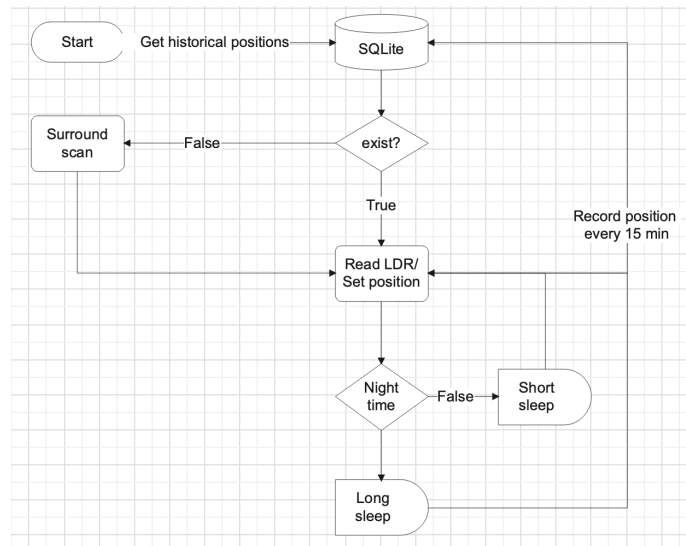


Figure 2.18 Flowchart of the DAST algorithm.

2.2.4 Servo Motor Controller

A separate thread controls each servo. Every thread receives angular positions from Master Controller calculated from DAST algorithm. The payload is a 2-byte unsigned integer pushed into a named pipe represents the rotational angle. Since mechanical components cannot execute the command fast enough and to avoid the overflow, we discard three out of four payloads within the pipe as shown in figure 2.19. Once the payload is received, it is converted to a PWM value and sent to the servo for execution.

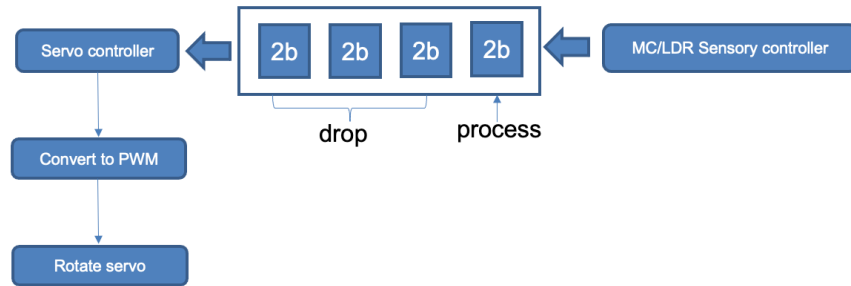


Figure 2.19 Flow of the Servo Motor Controller Thread.

To rotate the servo motor to certain degree, we convert the angular value to target duty cycle.

Conversion method is listed in equations 2.8

$$D_{DutyCycle} = \left[\frac{(T_{max} - T_{min}) \times (\theta_{target} - \theta_{min})}{\theta_{max} - \theta_{min}} + \theta_{min} \right] \times \frac{19.2 \times 10^6}{192} \quad (2.9)$$

Where T_{min} and T_{max} is the pulse width of the servo and θ_{min} and θ_{max} maximum and minimum rotational angles of the servo. These values are obtained during the initial calibration/setup of the system. Finally, θ_{target} is the target angular position of the servo which is calculated by the DAST algorithm.

2.2.5 Battery Controller

The battery controller is a reactive component responsible for the Li-Po battery. It was written using a PiJuice library. The battery Controller can get the charge level, current, and voltage of the battery. Additionally, it is responsible for enabling and disabling battery charging and setting the system wake up level when battery charge reaches the certain level.

2.2.6 Master Controller

Master controller (MC) is a main driver that operates all the components listed above. MC divides the system into four run levels based on Li-Po battery charge level.

Table 2.2 Description of Run Levels

Run Level	Battery Level	Action
GREEN	80% - 100%	No need to charge.
YELLOW	20% - 79%	No need to charge.
RED	5% - 19%	Charge if energy source is available.
BLACK	0% - 4%	Charge if energy source is available. Otherwise set wake-up level to 5% and perform system halt.

Upon startup, MC reads the configuration file to obtain all necessary attributes to perform general initialization of the system such as startup of attached devices, SQLite database, download sunrise and sunset data for the current day from weather.com, IPC initialization. Post initialization phase searches for the most optimal position to adjust the solar panel. Given that we store historical information of angular positions in our database, we take the average of the last three days of the current time and adjust the photovoltaic panel accordingly. Hence,

$$\bar{x} = \frac{1}{3} \sum_{t-3}^{t-1} x = \frac{1}{3} (x_{t-3} + x_{t-2} + x_{t-1}) \quad (2.10)$$

In case the optimal position has not been found, MC initiates a surround scan to locate the Sun by executing the LDR Sensory controller. Once Sun is located, the position is recorded in the database every 15 minutes. Earlier in LDR Sensory controller section, we mentioned a mysterious function `getSleepTime_BasedTimeOfDay()`. This function calculates the sleep time based on the sunrise and sunset data we download during the initialization phase of MC. Given that the sunrise and sunset times shift daily, download happens automatically every day between 00:00 and 01:00. Once data is downloaded and parsed, IziBu compares and calculates the adjustment angles for the Servo Controllers, which in turn are pushed to pipes to be read by Servo Motor Controller. Hence, photovoltaic panels will be ready to harvest Sun energy upon sunrise. In addition, MC controls the current flow from the photovoltaic panel to the ACS712 sensor. MC switches the relay module and redirects the current to UPS for Li-Po charging if the battery has dropped to RED run level. When the battery reaches 100% (GREEN run level), the relay is switched to NC and redirects the current to the current sensor for continuous current monitoring. As previously stated, we use ACS712-30A Hall-Effect current sensor, which outputs 66mV/A analog signal. For every 1A increase in current flow, we get a 0.066 V increase in the output signal. We have connected our sensor to the PCF8591 8-bit converter and used the conversion method

$$I = \frac{V - 2.5}{0.066} \quad (2.11)$$

Where I is the output current and V is the output from Equation 2.3.

By default, the output current from the photovoltaic panel is directed to UPS. This was done by connecting the NC terminal of the relay to UPS. Thus, if the Li-Po battery dies before sunrise, that is in BLACK run level and less than 5% charge remaining, IziBu has been programmed to auto-start the system once the battery charge level is back in the RED run level. The battery will remain to charge until it reaches 100%. Section 2.2.7 includes flowchart of the scheduler.

2.2.7 Flowchart

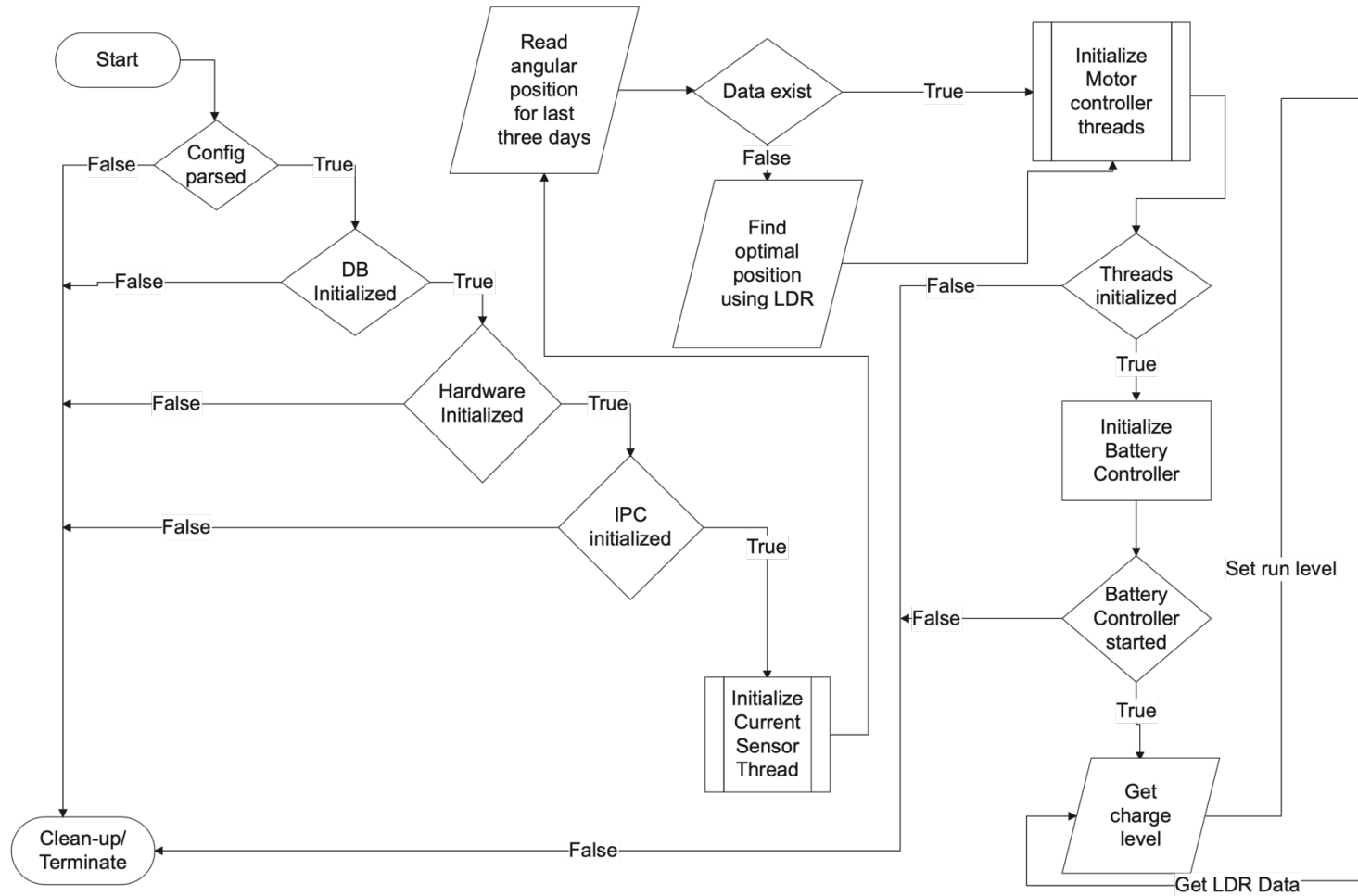


Figure 2.20 First part of flowchart.

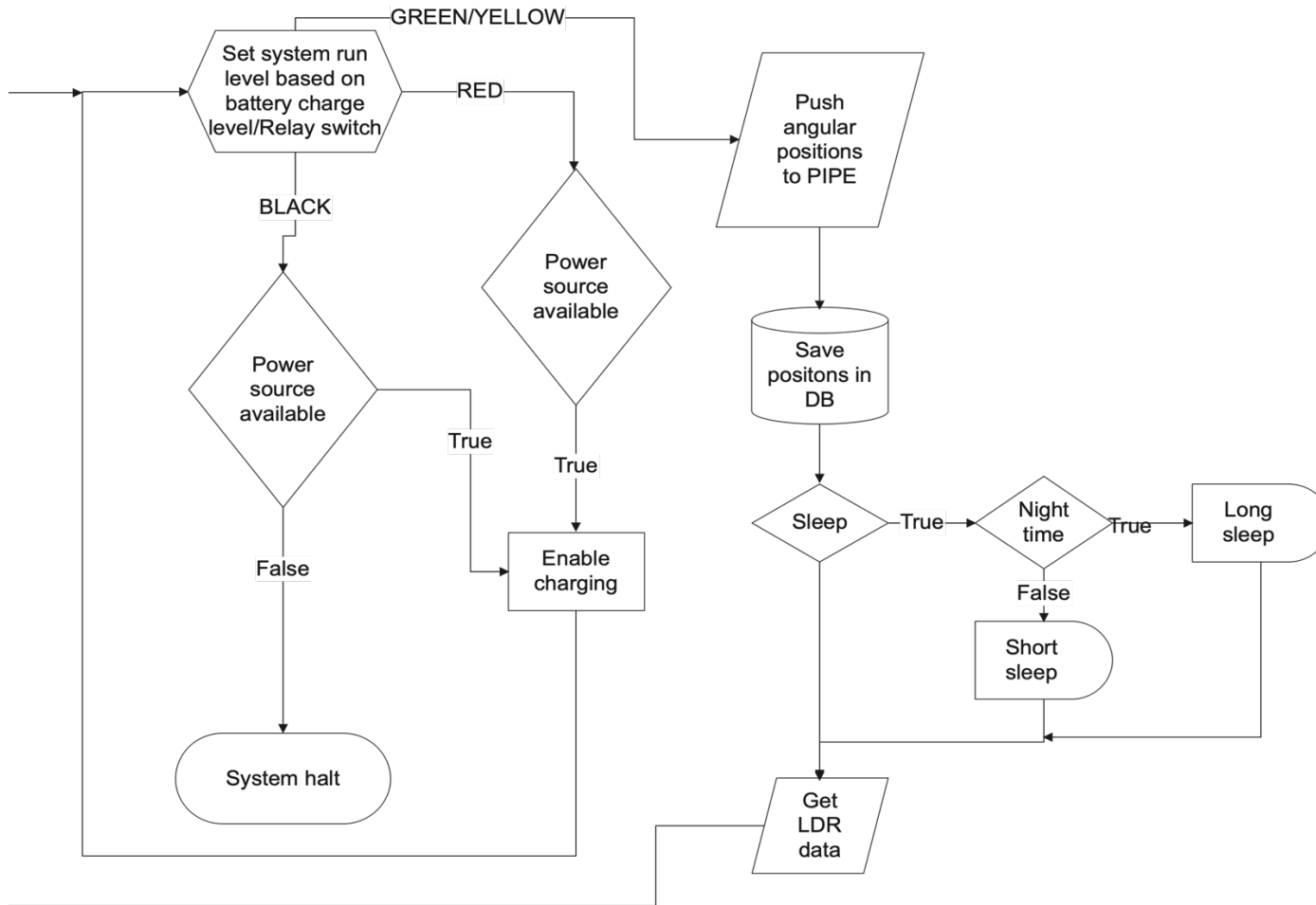


Figure 2.21 Second part of flowchart.

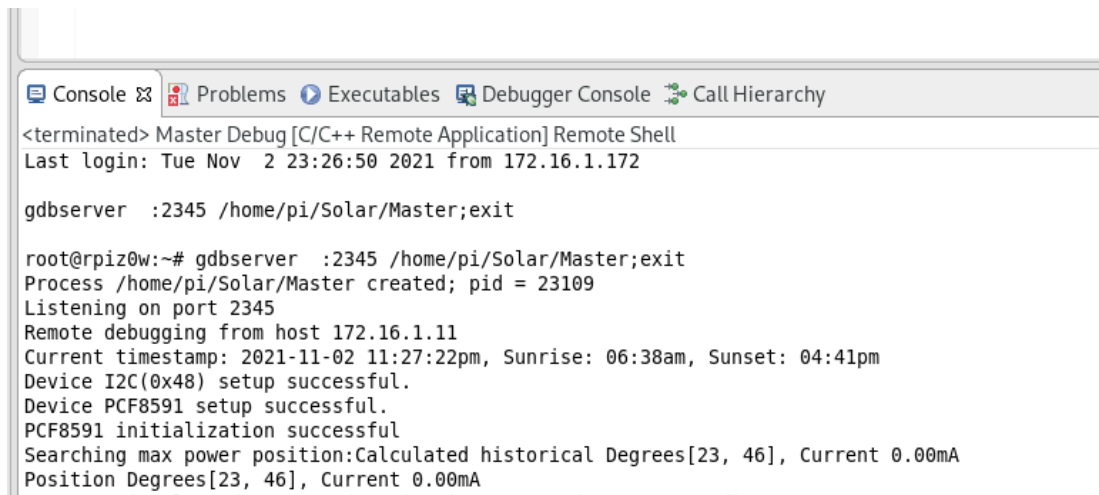
CHAPTER 3

EXPERIMENTAL RESULTS

In order to achieve experimental results and see how much power is supplied to an automaton, we have used a DC Power supply by Eventek Model KPS305D. It allows us to control voltage and current sent to IziBu.

3.1 Initialization and Setup

Figure 3.1 demonstrates the output results of the initialization phase of the scheduler. As shown, all attached devices have been initialized, sunrise and sunset data have already been downloaded, and the position for the next sunrise has been calculated.



```
<terminated> Master Debug [C/C++ Remote Application] Remote Shell
Last login: Tue Nov  2 23:26:50 2021 from 172.16.1.172

gdbserver :2345 /home/pi/Solar/Master;exit

root@rpiz0w:~# gdbserver :2345 /home/pi/Solar/Master;exit
Process /home/pi/Solar/Master created; pid = 23109
Listening on port 2345
Remote debugging from host 172.16.1.11
Current timestamp: 2021-11-02 11:27:22pm, Sunrise: 06:38am, Sunset: 04:41pm
Device I2C(0x48) setup successful.
Device PCF8591 setup successful.
PCF8591 initialization successful
Searching max power position:Calculated historical Degrees[23, 46], Current 0.00mA
Position Degrees[23, 46], Current 0.00mA
```

Figure 3.1 Screenshot of initialization.

3.2 Current and LDR Sensory Results

To simulate the power transfer of the photovoltaic panel to an automaton, we have set the voltage value to 5.2V, and 0.2 A. Supply current has been fluctuating between 0.1A to 0.2A.

Figures 3.2 and 3.3 demonstrates input from a DC power supply, LDR calculated values (A, B, C), angular value for every servo, and incoming current read from the power supply.

```

Console Problems Executables Debugger Console Call Hierarchy
<terminated> Master Debug [C/C++ Remote Application] Remote Shell
A=25, B=29, C=26, Degree[33:135] Current:0.16A
A=21, B=28, C=26, Degree[33:132] Current:0.16A
A=18, B=29, C=26, Degree[33:131] Current:0.16A
A=15, B=29, C=27, Degree[33:130] Current:0.16A
A=13, B=29, C=26, Degree[33:129] Current:0.16A
A=12, B=29, C=26, Degree[33:128] Current:0.16A
A=12, B=29, C=26, Degree[33:127] Current:0.16A
A=11, B=29, C=26, Degree[33:126] Current:0.16A
A=12, B=29, C=27, Degree[33:125] Current:0.16A
A=13, B=29, C=26, Degree[33:124] Current:0.16A
A=17, B=28, C=27, Degree[33:123] Current:0.16A
A=19, B=28, C=26, Degree[33:122] Current:0.16A
A=21, B=28, C=26, Degree[33:121] Current:0.16A
A=24, B=28, C=26, Degree[33:120] Current:0.16A
A=26, B=42, C=26, Degree[34:119] Current:0.14A
Panel(26,26,26) Power: current 13.81mA Sleep 10 seconds

```

Figure 3.2 Screenshot of Current and LDR Sensory results.

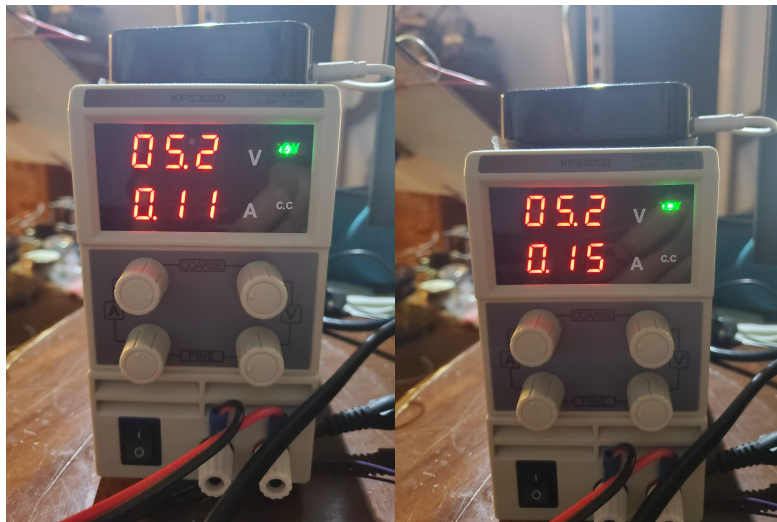
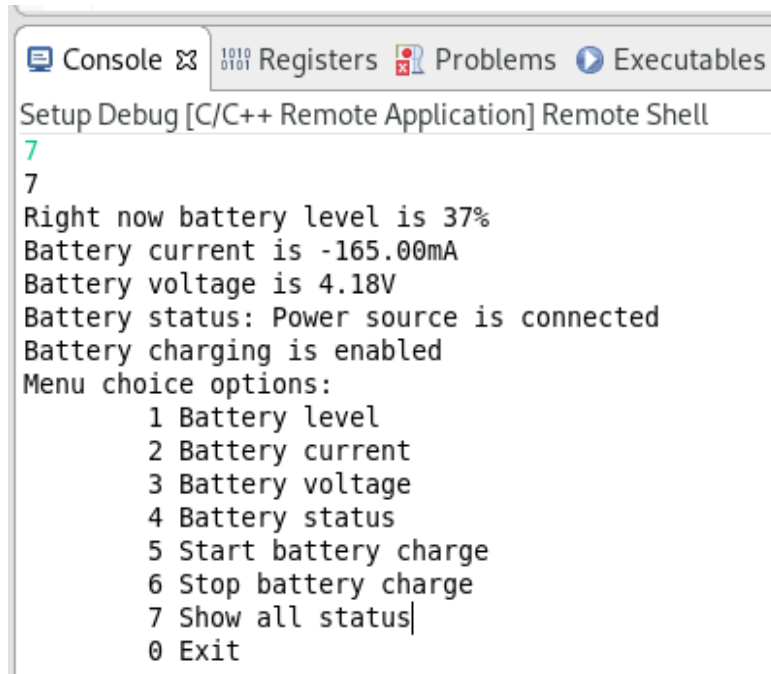


Figure 3.3 DC power supply.

3.3 Li-Po Battery Management

Figure 3.4 demonstrates the battery management interface we integrated into IziBu. The management interface is part of a Setup and Calibration component of the IziBu. The current selection in the screenshot is #7, which exports all battery info. Additionally, battery charging can be enabled or disabled.



```
Console Registers Problems Executables
Setup Debug [C/C++ Remote Application] Remote Shell
7
7
Right now battery level is 37%
Battery current is -165.00mA
Battery voltage is 4.18V
Battery status: Power source is connected
Battery charging is enabled
Menu choice options:
  1 Battery level
  2 Battery current
  3 Battery voltage
  4 Battery status
  5 Start battery charge
  6 Stop battery charge
  7 Show all status|
  0 Exit
```

Figure 3.4 Screenshot of Li-Po battery management component.

3.4 Autonomy

The last experiment we conducted was the autonomy of the IziBu. We charged the battery to 95%, disabled battery charging, and disconnected the power supply. Then we retrieved battery charge levels in 5-minute intervals. Given that our battery capacity is only 500mAh, we were able to reach 66 minutes of autonomy. Autonomy can be improved by connecting a larger battery.

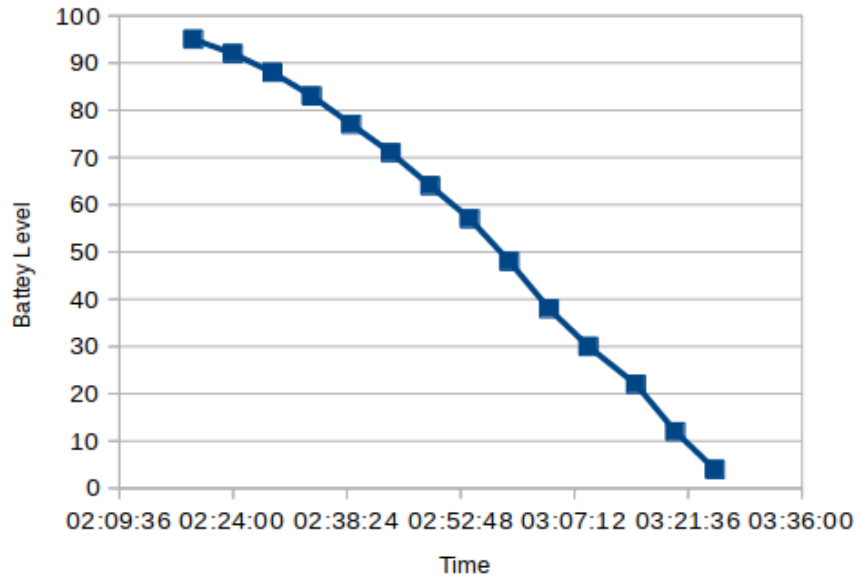


Figure 3.5 Battery discharge graph.

CHAPTER 4

CONCLUSION

In this thesis, we introduced the base design of photovoltaic energy harvesting automaton IziBu. The objective of this study was to design and implement a complete application of the off-grid, scalable system with autonomy functionality. The system consisted of hardware and software components. The hardware component includes a single board computer with Linux OS, current sensor, photoresistors for light intensity detection and DAST implementation, relay module to control the current flow, and analog-digital converter. The software component has been designed to be event-driven, reactive and includes multiple subcomponents to utilize hardware devices and a general logic that completes the automaton.

REFERENCES

- [1] Enerdata. (n.d.). Electricity domestic consumption. Retrieved from Enerdata: <https://yearbook.enerdata.net/electricity/electricity-domestic-consumption-data.html>
- [2] Hall effect sensor. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Hall_effect_sensor
- [3] Hattersley, L. (2019). Raspberry Pi 4, 3A+, Zero W - specs, benchmarks & thermal tests. Retrieved from MagPi Magazine: <https://magpi.raspberrypi.org/articles/raspberry-pi-specs-benchmarks>
- [4] IEA. (2020). Solar PV. Retrieved from International Energy Agency: <https://www.iea.org/reports/solar-pv>
- [5] PiJuice. (n.d.). Retrieved from GitHub: <https://github.com/PiSupply/PiJuice>
- [6] Raspberry Pi. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Raspberry_Pi
- [7] Rush, C. (2019, March 26). How to save Power on your Raspberry Pi. Retrieved from Pi-Supply: <https://learn.pi-supply.com/make/how-to-save-power-on-your-raspberry-pi>
- [8] <https://us.sunpower.com/sites/default/files/media-library/white-papers/wp-understanding-solar-system-ratings.pdf>
- [9] El Hammoumi, Aboubakr & Motahhir, Saad & Abdelaziz, el ghzizal & Chalh, Abdelilah & Derouich, Aziz. (2018). A simple and low-cost active dual-axis solar tracker. Energy Science & Engineering. 6. 10.1002/ese3.236.
- [10] Iqbal, Tariq. "IoT Based Renewable Energy Management and Monitoring System for the Frist Passive House in Newfoundland." NECEC (2020): n. pag. Print.

[11] Victor, João & Juca, Sandro & Pereira, Renata & Carvalho, Paulo & Fernández-Ramírez, Luis. (2019). IoT Monitoring systems applied to photovoltaic generation: The relevance for increasing decentralized plants. *Renewable Energy and Power Quality Journal*. 17. 536-545. 10.24084/repqj17.368.

[12] Layse Nascimento. “Internet of Things-Aided Smart Home Off-Grid Photovoltaic-Powered.” *Internet of Things-Aided Smart Home Off-Grid Photovoltaic-Powered* (2020): n. pag. Print.

[13] Aghenta, Lawrence Oriaghe and M. Tariq Iqbal. “A Low-Cost, Open Source IoT-Based SCADA System Design, and Implementation for Photovoltaics.” (2019).

[14] Solar Reviews. (2021). Retrieved from Solar Reviews:

<https://www.solarreviews.com/blog/what-are-the-most-efficient-solar-panels>

[15] Maeng, Kiwan & Lucia, Brandon. (2020). Adaptive low-overhead scheduling for periodic and reactive intermittent execution. 1005-1021. 10.1145/3385412.3385998.

[16] Jeff Geerling. (2015). Retrieved from Jeff Geerling:

<https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-power>