Chapter 1

Processing and Control

Prepared by Si Teng Wu - WXXSIT001

1.1 Introduction

This subsystem involves designing a hardware solution and software for controlling the collection, processing, storage, and transmission of data, as well as an identification solution. It has been divided into the following submodules: Identification, Weight Processing, Storage, Transmission, System Control, and Power Saving Techniques. Through the use of a microcontroller module and various other hardware modules, this subsystem will be the 'brain' of the system and act as the central hub for data while the system is in the field. This subsystem will receive input data from the Scale and Power subsystems and provide output data to the UI subsystem.

1.2 Requirements and Specifications

User Requirements

The stakeholder/user, Carrie Hickman, a PhD student at the Fitzpatrick Institute of African Ornithology, UCT, has provided the following user requirements that are releveant to this susbsystem. These requirements were collected through stakeholder engagements on the EEE4113F Teams [1].

- The bird being weighed needs to be identified.
- The weight should be accurate to within a gram.
- There should be some way to collect data without climbing the tree.
- The data should be stored on an SD card as a backup.
- The system should last for two weeks.

Requirements Analysis

Using these user requirements, the following functional requirements (FR), specifications (SP), and Acceptance Test Procedures (ATP) were created. The specifications provide a standard for the subsystem, and the ATPs will be used to test the standards of the proposed solution.

FR ID	Functional Requirement
FR-1	Control the system using a microcontroller.
FR-2	Use a suitable identification technique on the bird as it is being weighed.
FR-3	Collect diagnostic battery data.
FR-4	Collect the weight data.
FR-5	Develop a weight processing algorithm that can process the weight data.
FR-6	Store the data on local, non-volatile and removable storage.
FR-7	Transmit the data wirelessly from the nest to the user's device on the ground.
FR-8	Use power saving techniques to limit the use of power.

Table 1.1: Functional Requirements of the subsystem

SP ID	Specification	FR ID	ATP ID
SP-1	Use the ESP32 S3 Dev Module operating at 3.3V and <250mA.	FR-1	ATP-1
SP-2	Use the RDM6300 125kHz RFID module operating at 5V <50mA.	FR-2	ATP-2
SP-3	The RFID must detect PIT tags at a minimum distance of 5cm and use the GPIO pins to switch on/off.	FR-2	ATP-3
SP-4	Activate RFID only when the weight detected exceeds 2.2 kg.	FR-2, FR-8	ATP-4
SP-5	Use the ADC to read the normalised battery level (0-3.0V) and calculate it as a percentage.	FR-3	ATP-5
SP-6	Read the HX711 using I2C every 10 seconds.	FR-4, FR-8	ATP-6
SP-7	Implement a suitable filtering technique (normal average, moving average, exponential average, or median filter) to process the weight data.	FR-4	ATP-7
SP-8	Utilize a micro SD card module connected via SPI to store data in a text file format <weight,ID,battery%>.	FR-6	ATP-8
SP-9	Implement a web server over WiFi to transmit data.	FR-7	ATP-9
SP-10	Utilize deep sleep mode to minimize power consumption whenever possible.	FR-8	ATP-10
SP-11	Use a 433MHz remote transmitter and receiver to start the WiFi when necessary.	FR-7, FR-8	ATP-11

Table 1.2: System Specifications and associated FRs and ATPs

The following ATPs use code that is on GitHub under the 'main' directory via this link: GitHub.

ATP ID	Test Procedure	Success Criteria
ATP-1	Setup a power supply to 3.3V and connect the positive lead to the 3.3V pin and negative lead to GND pin on the ESP32 S3 (MCU). Flash the code Test_ESP32S3_Power.ino and observe the current draw and onboard LED.	The ESP32 S3's red LED is on and the current draw is <250mA.
ATP-2	Setup a power supply to 5.0V and connect the 5V and GND of the RDM6300. Observe the current draw and onboard LED.	The LED is on and the current draw is <50mA.
ATP-3	Connect the TX pin of the RDM6300 to MCU pin 4 and flash Read_RFID.ino. Power both modules (ATP-1, ATP-2), and connect the MCU to a PC. Bring an RFID tag close to the antenna and observe the serial console. Measure the maximum distance of detection.	Distance is greater than 5cm.
ATP-4	Connect the RFID module, HX711, and micro SD module to the MCU (ATP-1,3,6,8). Flash Final.ino, hover a tag above the RFID antenna and place a 2 kg and 5 kg weight, separately.	The console only outputs 'Weight detected' and reads RFID when using 5kgs.
ATP-5	Connect the power supply to MCU pin 7 and flash PollingADC-BATT.ino. Change the voltage to 0V, 1V, 2V, 3V and observe the console.	Console outputs 0, 33, 66, 100% at respective levels.
ATP-6	Setup a power supply at 5V and connect the HX711 5V and GND pin to the supply and DAT and CLK to MCU pin 8 and 9, respectively. Flash WeighingScale.ino, place a weight on the scale and observe the serial console.	Console shows weight data.
ATP-7	Run the filtering techniques on clean, noisy and bad data, and compare performance.	<5% error on clean and noisy data and <10% accuracy on bad data.
ATP-8	Setup a power supply at 5V and connect the 5V and GND of the micro SD card module. Connect pins CS, MOSI, SCK, MISO to MCU pins 10, 11, 12, 13, respectively. Flash SDTest.ino and press reset on the ESP32S3. Remove the SD card and check its contents.	There exists a file, Test.txt, with 'Hello World!' in it.
ATP-9	Connect the remote module to the MCU (ATP-1,11) and extend the antenna on the remote. Flash WifiSend.ino, press button B on the remote, connect to 'ESP32-Access-Point', using password '123456789' on a user device and run the python script GetrRequest.py.	A text file is downloaded in the directory with 'Hello World!' in it.
ATP-10	Connect the remote module to the MCU (ATP-1,11) and extend the antenna on the remote. Flash Sleep.ino, press button B on the remote and observe the serial console.	The console shows the ESP32 resetting every 10 seconds, and resets and prints 'Awake!' upon button press.
ATP-11	Setup a power supply at 5V and connect the 5V and GND pins. Connect a voltage divider using a 4.7k and 12k resistor to GND from pin D0. Connect the output of the voltage divider to MCU pin 6. Power the MCU (ATP-1) and flash WiFiSend.ino. Press the button B and check on a user device for WiFi networks.	'ESP32-Access-Point' is present on WiFi networks.

Table 1.3: Acceptance Test Procedures

1.3 Design Choices

Microcontroller

This susbsystems needs a microcontroller to act as the central processing unit. There are various options on the market, but the design of a module to interface with a microcontroller IC is not within the scope of this project so the decision was made to use a off-the-shelf microcontroller kit/dev module. The Raspberry Pi Pico W, ESP32 series and Arduino Nano dev boards were considered. The cost, availability, features and to a lesser extent, software support, were considered when the decision was made. In terms of features, WiFi, flash storage, communication peripherals, deep sleep mode and clock speed were considered.

Microcontroller	Price	WiFi	Flash	SPI,I2C,UART	Sleep	Clock Speed
Pi Pico W	R150.00	Yes	2MB	Yes	Yes	133MHz
ESP32 S3	R178.00	Yes	16MB	Yes	Yes	240MHz
Arduino Nano	R200.00	No	32kB	Yes	Yes	16MHz

Table 1.4: Comparison of Microcontrollers [2][3][4]

It was decided, based on the results shown in Table 1.4, that the ESP32 S3 Devkit C was the most suitable choice for this design. The price is in the middle of the options, but has all the features, 16MB of flash, allowing for more space for instructions, and the highest clock speed, allowing for faster processing. The Pi Pico Wcould have been a viable option, but lacks in flash space and clock speed compared to the ESP32 S3. The Arduino Nano AT328P does not have onboard WiFi so it would not be a viable option.

In terms of choosing the ESP32 S3, it was a trivial decision based off of flash, price and avaialblity. The ESP32 prices ranged from R150 to R240, approximately, but the S3 boasts the best features at a mid range price. It was also in stock and available. Finally, the ESP32 S3 has many software libraries and interfaces with Arduino IDE.

Identification

The identification options were limited. The user is currently able to identify the birds using a camera trap (visually) and by bird calls (accoustically). Due to the voilent nature of the Southern Ground Hornbill, introducing hardware elements outside of the housing would be dangerous for both the hardware and the bird, so an external camera was not a viable solution. Recording of the birds while weighing the is possible, but this poses two problems. The storage space required to store audio files would be much than just text based data and more importantly, there is no guarantee that the bird will do its call when its is being weighed. The RFID solution was then proposed to the stakeholder, who agreed that its would be possible to tag the birds when they are young [1]. RFID would allow for the birds to be consistently identified when they are weighed, uses low storage to store the ID, and can be done without exposing hardware to the extrenal environments or the birds.

There are three possible types of RFID that could be used. Low frequency (LF), high frequency (HF) and ultra-high frequency (UHF). The power consumption, size, range and avaialblity of Passive

Integrated Transponders (PIT) tags for animals were considered for the decision.

Frequency	Operational Range	Power Consumption	PIT Tag Availability
LF (125-135kHz)	Up to 20 cm	Low (0.25W)	Common (125kHz or 134.2kHz)
HF (13.56MHz)	Up to 20 cm	Slightly higher	Rare for animal PIT tags
UHF (868-928MHz)	Up to 3 m	Significant	Limited availability

Table 1.5: Comparison of RFID Frequencies [5]

As shown in Table 1.5, the most suitable solution for a low power design would be the LF, as it uses the lowest power consumption and has the most readily availbale supply of animal PIT tags. For this design, an 125kHz RFID reader module, RDM6300, was chosen and common proximity tags were used to demonstrate and test the effectiveness of the RFID reader.

RFID Wiring and Control

Having chosen the RDM6300, and ESP32 S3, a challenge arose as to how to connect these modules effectively. The platform for the scale will be appoximately $1m^2$. Assuming the antennas for RDM6300 are designed to be 25cmx25cm squares in a future iteration, that would require 16 antennas to cover the surface of the scale. Due to the limited number of UART peripherals on the ESP32 S3, a choice needed to be made on how to effectievly have 16 antennas and one input for the ESP32. The following choices were considered:

- Uses a MUX for the antennas and a single RDM6300 module.
- Uses a MUX for 16 RDM6300 modules with indivdual antennas.
- Uses 16 RDM6300 modules and programically switch them on/off.

The first option was rejected as the antennas are tuned to 125kHz with specific Inductance-Resistance-Capacitance (LRC) values. A mux between the antennas and the module may cause parasitics that would change the resonance frequency of the antennas.

Both the second and third options are viable solutions, but the last solution was chosen. The ESP32 S3 has an abundance of GPIO pins and by turning each module on one at a time, this limits the peak current draw for the system, and ensures that no modules are drawing power while idle, waiting for their turn on the MUX. This also removes the need for an extra component, decreasing complexity. The Tx pin of all 16 modules would be connected to a single ESP32 Rx pin (pin 4). For this design, 3 modules were tested instead of 16 due to cost constraints.

WiFi vs Bluetooth

WiFi and Bluetooth were considered for the transmission of data from the device in the nest to the users device on the ground. It is expected that the trees for the nest can be up to 30m in the air and the device will be housed inside a wooden boox, on top of a nest, on top of a tree, therefore line of sight will not be available. The following factors were considered: Power consumption, max one-way data rate and nominal range.

Technology	Power Consumption (mA)	Data Rate (Mb/s)	Range (m)
WiFi	100-350	31.4	100
Bluetooth	1-35	0.732	10

Table 1.6: Comparison of Wireless Technologies [6]

It is evident from Table 1.6 that WiFi is the only solution, albeit more power hungry, that can reach the ranges required for this design. It is also noted that these ranges were tested in line of sight conditions, further cementing that WiFi is the better option.

Power Saving Considerations

These power saving choices were made in order to address the two week duration requested by the stakeholder. This subsystem will contain the majority of the elctronics used in the entire system, so it is very important to save power whever possible. The following choices and their reasons were made:

Component	Decision	Reason
RFID	Control the power for the RFID modules	Removes their power consumption completely when not being used.
RFID	Only use RFID when weight is above 2.2kgs	Ensures RFID is used only when a potential adult has landed.
Collection	Takes a single reading and waits for bird to leave before taking another	Ensures that a single bird will not be weighed and identified constantly when it is on the scale.
WiFi	Introduce a remote transmitter receiver that is used to start the WiFi download process	As the user only visits the nest fortnightly, the WiFi, which consumes the most power, will be off until the user manually turns it on.
WiFi	Turn off the WiFi if no one has connected after 30 seconds	Ensures the WiFi isn't left on if the user misclicks the remote.
Sleep	The ESP32 will sleep and wake every 10 seconds to take a sample reading or wake upon remote trigger for WiFi	Ensures that the ESP32 will draw minimum power while idling.

Table 1.7: Power Saving Decisions

1.4 System Design

Block Diagram and System Control Flowchart

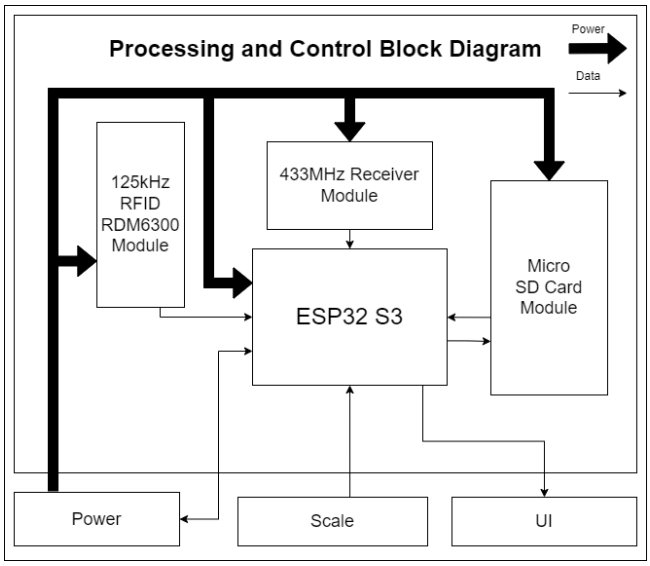

Figure 1.1: Block Diagram

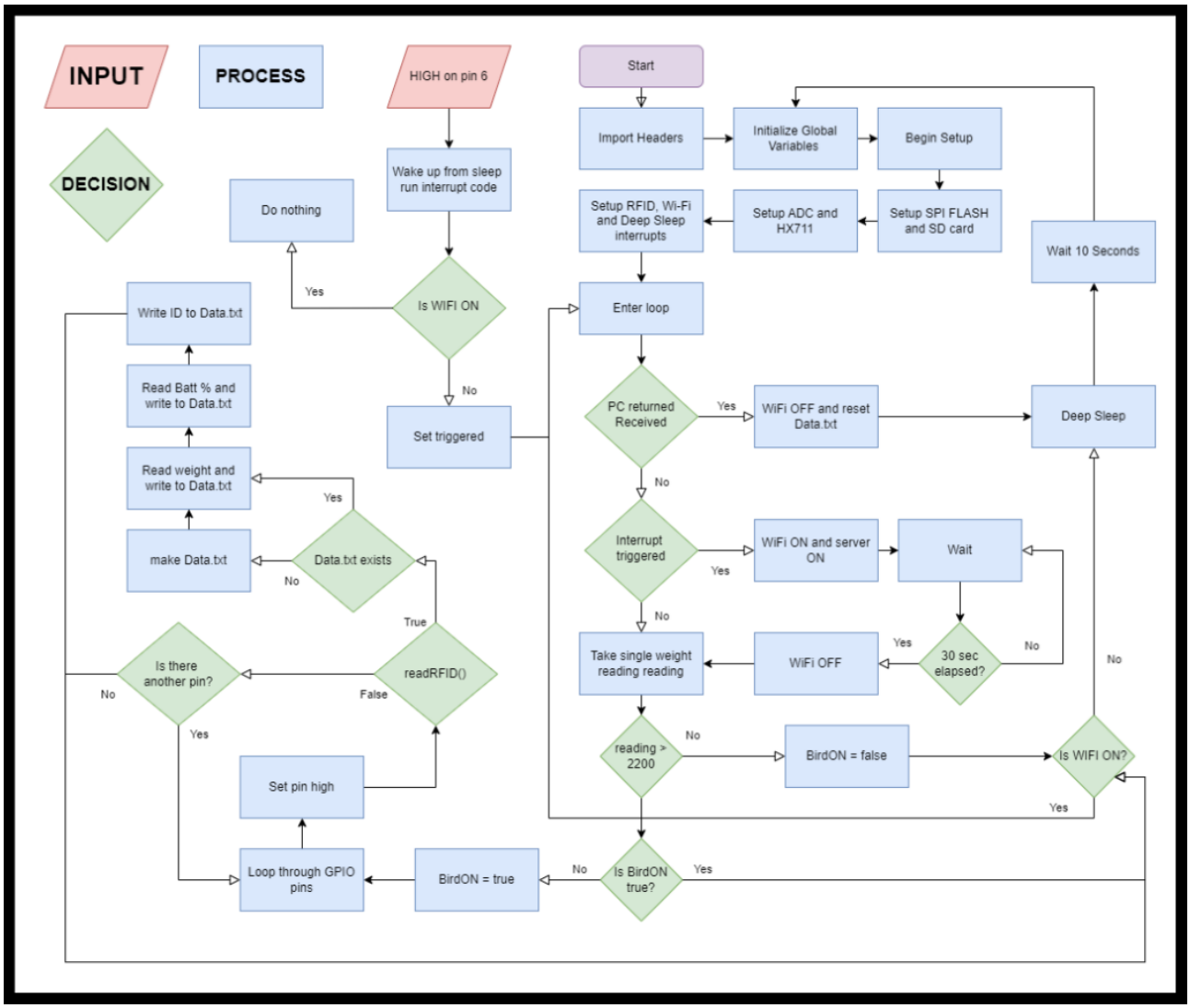

Figure 1.2: System Control Flowchart

Circuit Diagram

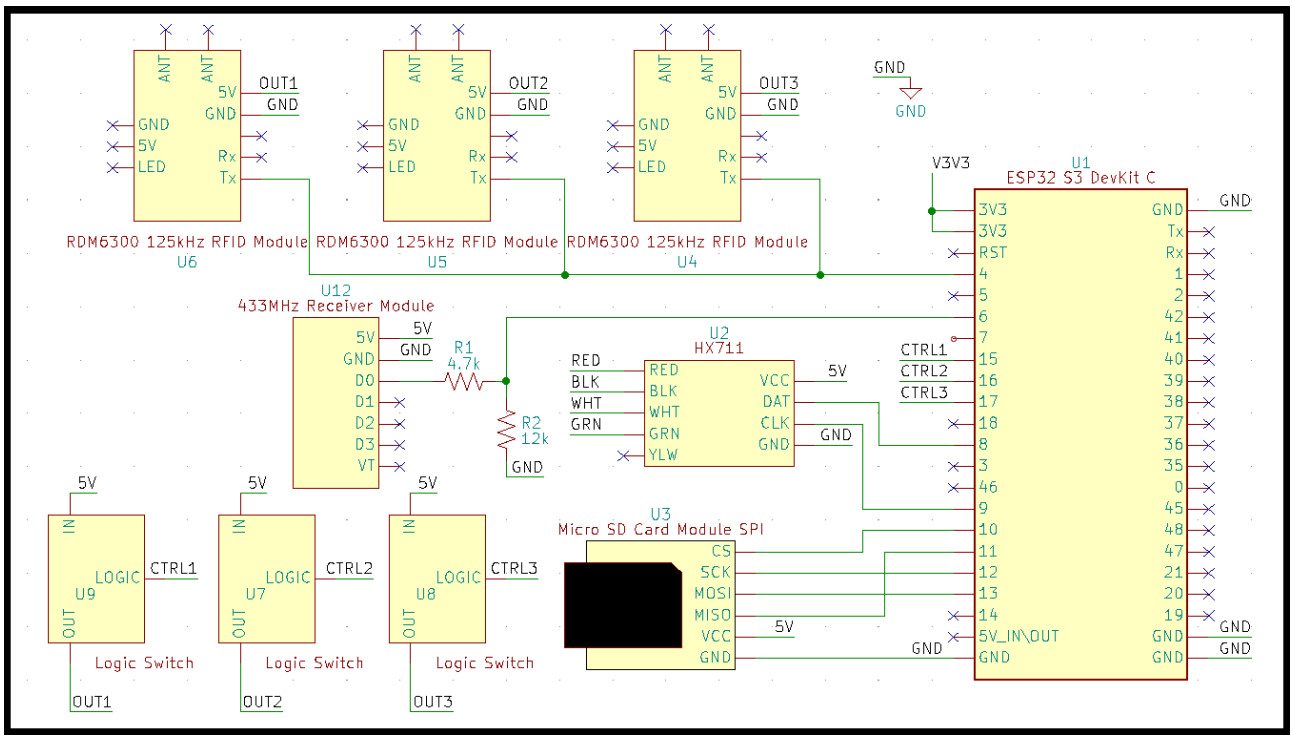

Figure 1.3: Circuit Diagram

Used Libraries

The following libraries are not part of the stand C libraries or standard Arduino libraries, but were used in the final code. Table **??**

Header File	Description	Author
HX711.h	Description of HX711.h	Author A
AsyncTCP.h	Description of AsyncTCP.h	Author B
ESPAsyncWebServer.h	Description of ESPAsyncWebServer.h	Author C

Table 1.8: External Libraries

Custom RFID Reading

Weight Processing Results

1.5 ATP Results

1.6 Conclusions and Recommendations

The Processing and Control subsystem contains the central processing unit of the entire system and acts as a cnetral hub for data while in the field. In this iteration, weight data and RFID data was

succesfully collected, stored in a micro SD card and upon an external remote button press, transmitted to the user device via WiFi. Weight processing techniques were investigated and it was found that the median filter performs best.

Bibliography

[1] EEE4113F 2024 - Stakeholder Engagement. All Carrie Hickman Posts. Microsoft Corporation. 12/05/2024. [Online]. Available: https://teams.microsoft.com/l/channel/19%3A16092253e43b43d0ae12e2f4b0cbdb59%40thread.tacv2/Stakeholder%20Engagement?groupId=d4192a4a-a750-48ea-bd8e-ee5cbd5776cb&tenantId=92454335-564e-4ccf-b0b0-24445b8c03f7&ngc=true

[2] Arduino, "ATmega328P Arduino Compatible Nano V3 Improved Version — pishop.co.za," https://www.pishop.co.za/store/arduino-boards/geekcreit-atmega328p-arduino-compatible-nano-v3-improved-version, 2024, [Accessed 12-05-2024].

[3] MicroRobotics, "ESP32 S3 Dev Board - 16MB Flash, 8MB PSRAM — robotics.org.za," https://www.robotics.org.za/ESP32-S3-N16R8?search=esp32, 2024, [Accessed 12-05-2024].

[4] PiShop, "Raspberry Pi Pico WH with pre-soldered headers — pishop.co.za," https://www.pishop.co.za/store/raspberry-pi-boards/raspberry-pi-pico-wh-with-pre-soldered-headers, 2024, [Accessed 12-05-2024].

[5] B. Fennani, H. Hamam, and A. O. Dahmane, "Rfid overview," in *ICM 2011 Proceeding*, 2011, pp. 1–5.

[6] E. Ferro and F. Potorti, "Bluetooth and wi-fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005.