Author: *Logan HB Becker*

Plant Health Monitoring and Notification System

Table of Contents

Overview

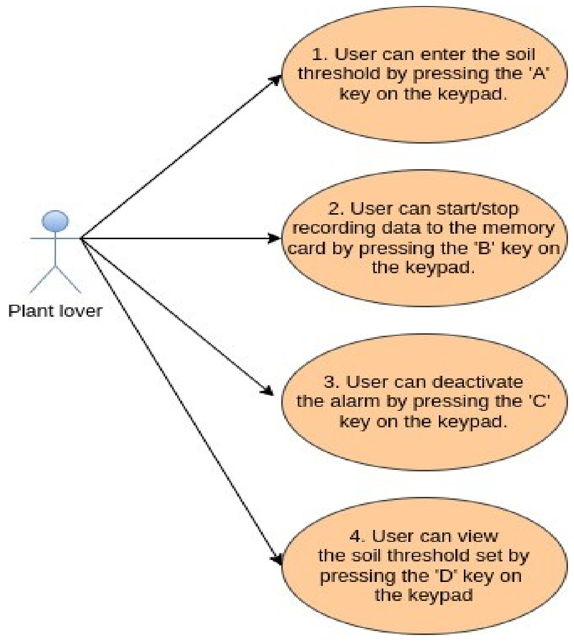
The objective of the project is to monitor and notify a user of the soil moisture levels in a plant. The moisture levels are recorded to an external memory card for analytical purposes. The user is provided with a keypad to enter the water threshold. The user has 1 option to receive notifications locally with an alarm system.

Problem and Motivation

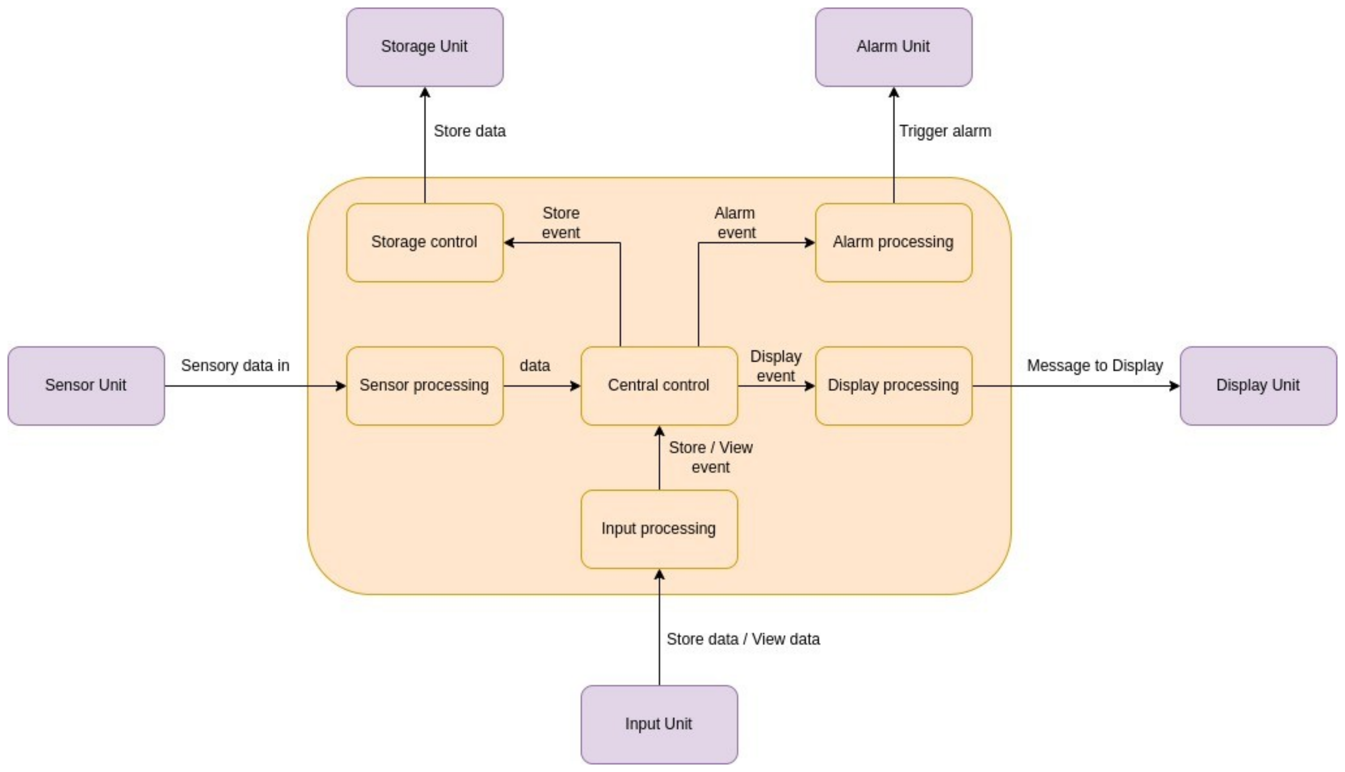
My brother grew an interest in planting, and at times he either forgets or is not home for about a week. The plant requires constant care such as regular watering, and being placed outside for fresh air and quality sun exposure. The inspiration to solve this problem came to my attention and I began to look at what components I had lying around, and how I could put them together to build a system to solve this problem for my brother.

1. Use cases

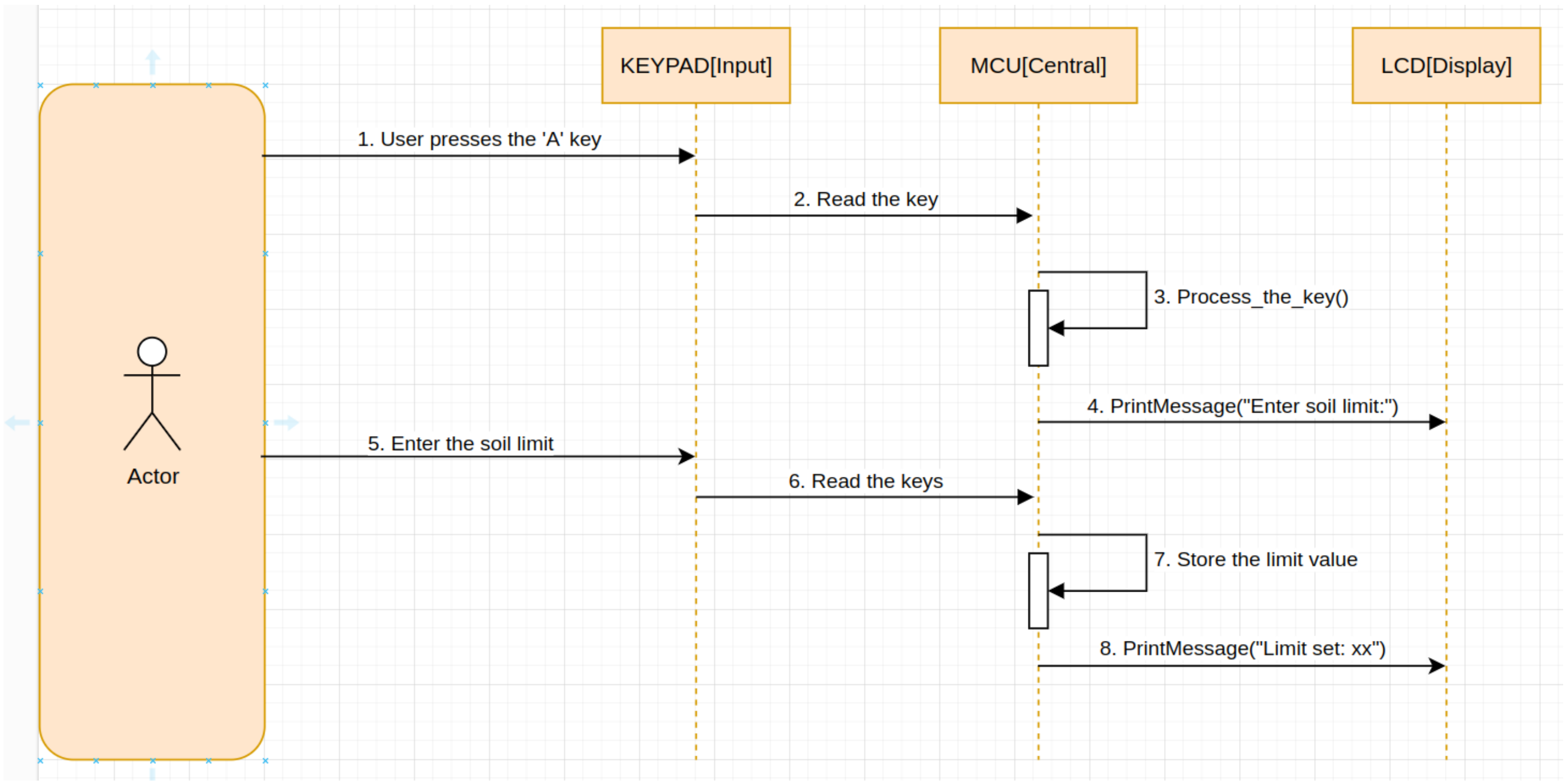
The use case diagram illustrates the high level view of the system from the perspective of the user. It is an ideal representation of *what the system* must be able to accomplish based on user inputs and its expected outputs. The same use cases can be used to replicate the system in a non-embedded manner by simply implementing these use cases in other programming languages such as Python or C. The possibility of running the system in a non-embedded manner by implementing and running the system on your local machine allows us to generate unit and integration tests that can effectively be utilized on the real hardware during the refactoring phase to catch or prevent bugs from being injected. It can further be utilized to generate **acceptance** test plans to ensure that all user requirements are achieved.


Plant lover
1. User can enter the soil threshold by pressing the 'A' key on the keypad.
2. User can start/stop recording data to the memory card by pressing the 'B' key on the keypad.
3. User can deactivate the alarm by pressing the 'C' key on the keypad.
4. User can view the soil threshold set by pressing the 'D' key on the keypad

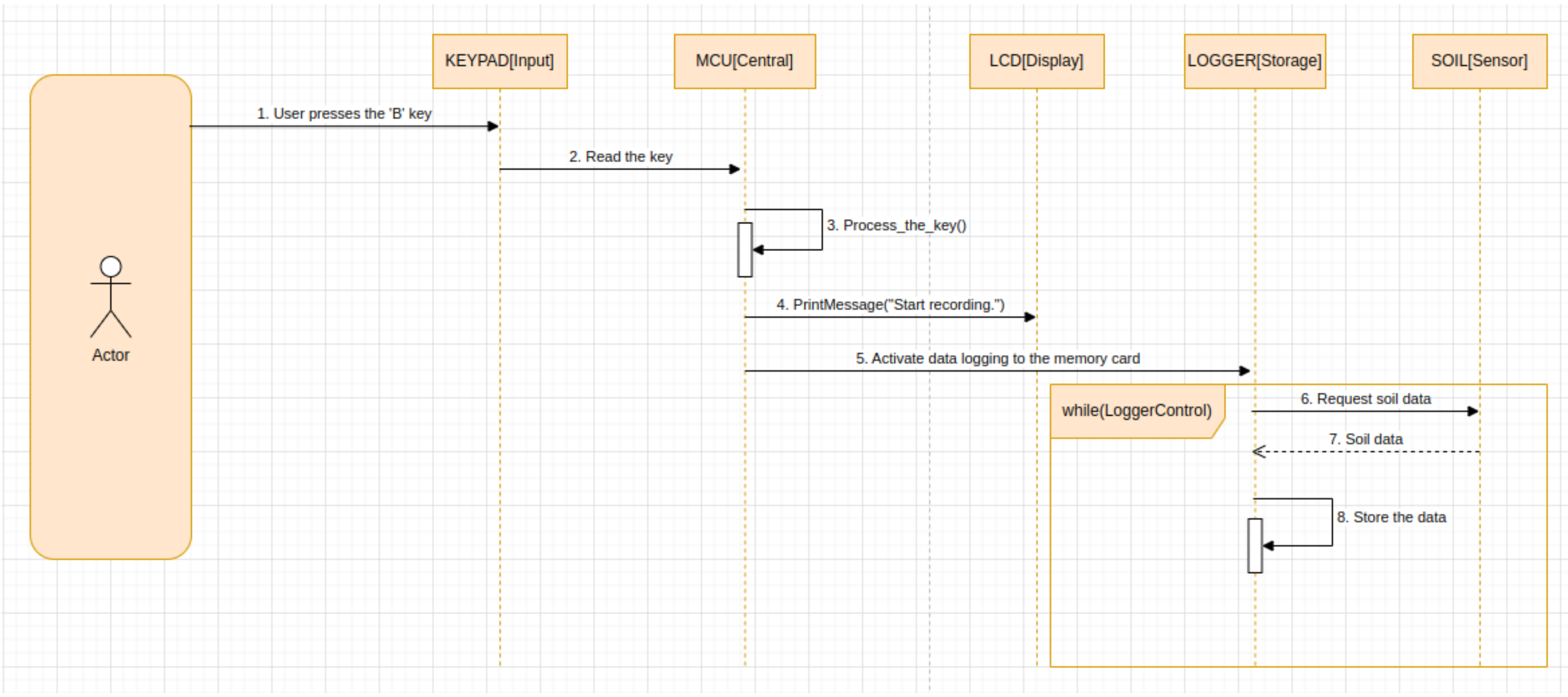
2. Architecture (High Level Design)



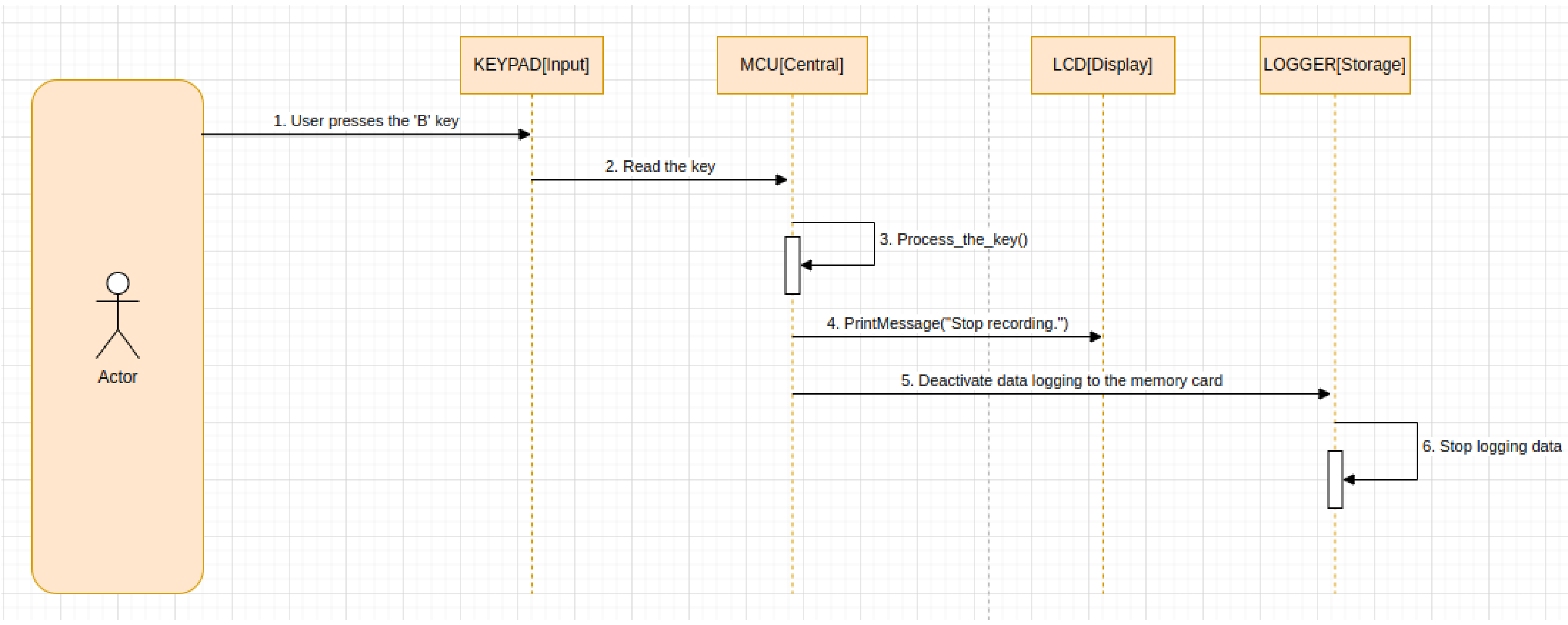
2.1. When the user presses ‘**A**’ on the keypad, then it will notify the **Central control** to send a message “***Enter soil limit***” to the **Display Unit**. The user now enters the **soil limit** and then presses the **‘#’** to confirm the value. Otherwise, they can press **‘*’** to cancel the operation.



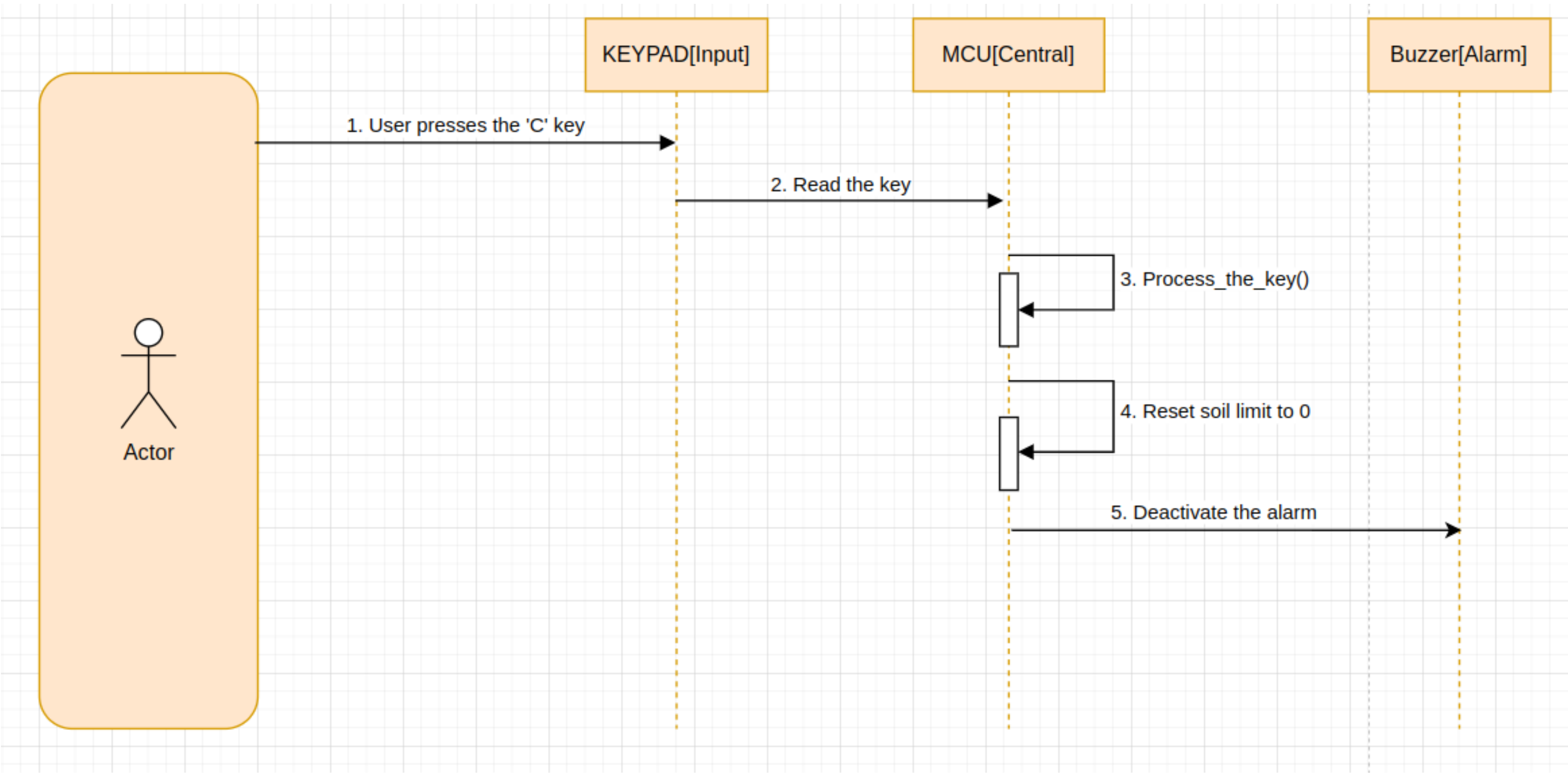
2.2.1. When the user presses ‘**B**’, then a display message of “***Start recording***” is sent to the **Display unit** and the onboard led stops blinking, then the **Central control** will initiate a message to the **Storage unit** to store the soil humidity data an interval of 1 second.



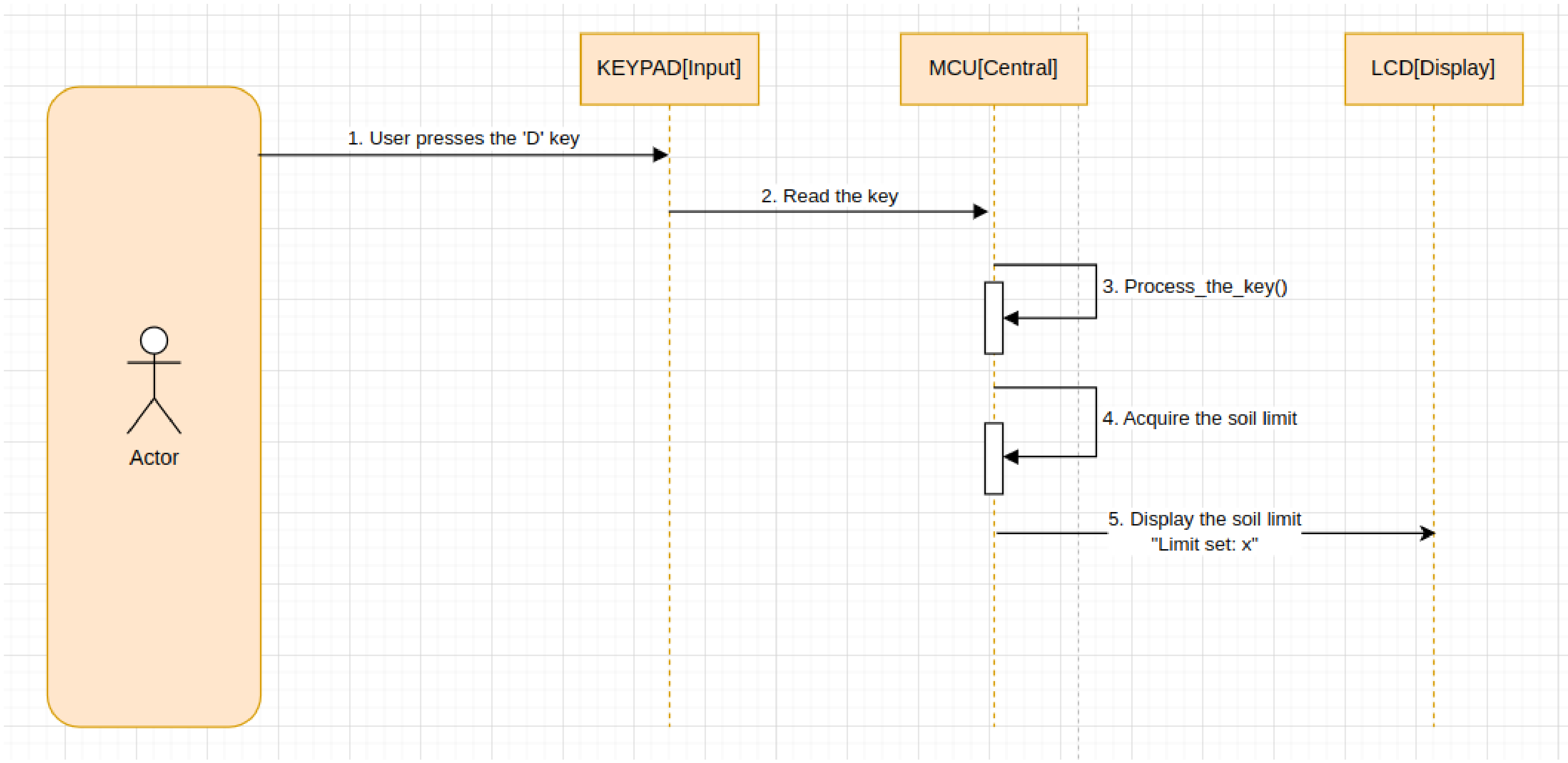
2.2.2. When the user presses ‘**B**’, then a display message of “***Stop recording***” is sent to the **Display unit** and the onboard led starts blinking, then the **Central control** will initiate a message to the **Storage unit** to stop recording the soil humidity data.



2.3. When the user presses ‘**C**’ on the keypad, then it will notify the **Central control** to clear the **soil limit** value, and then it will send a deactivate alarm event to the **Alarm unit.**



2.4. When the user presses ‘**D**’ on the keypad, then it notify the **Central control** to send a message “***Limit set: x***” to the **Display unit** to show the user the limit set.


KEYPAD[Input]
MCU[Central]
LCD[Display]
Actor
1. User presses the 'D' key
2. Read the key
3. Process_the_key()
4. Acquire the soil limit
5. Display the soil limit "Limit set: x"

3. Analysis of the System

A system can be simple or extremely complex, therefore the best approach in this scenario is to always analyze the system by **understanding what it needs to do** and with that to **extract** the **components** required to make the system work.

3.1 What does the system need to do? **(understand)**

1. The user must be able to **monitor** the soil moisture level of a pot plant.
2. The user must be able to **input** the soil moisture threshold into the system.
3. The user must be able to be **notified** by the system when the soil moisture threshold is triggered.
4. The user must be able to **store** the data to an external memory card.
5. A sensor must be present to **measure** the soil moisture in the pot plant
6. All the inputs/outputs must be consumed/produced by a **system**.

Now that a more detailed understanding of "what" the system needs to do is known.
What components are required to achieve the system goals? (extracting)

3.2 What components are required to achieve the system goals? (**extract**)

1. A **display** is required such as a 7 segment display or OLED or LCD or LED'S or a touchscreen.
2. An **input** is required such as buttons or a keypad or a touchscreen.
3. An **alarm** is required such as a buzzer to notify by sound or 2G module to notify by an SMS or WIFI / BLE to send notifications to a Mobile App.
4. A storage device is required to **store** the data such as an memory card.
5. A soil moisture sensor for **measuring** is required such as keyestudio soil humidity sensor.
6. A **processing** unit is required such as digital/analog circuits, microprocessor or a microcontroller (e.g. MSP432xx, PIC32xx, STM32xx, ESP32xx, NRF51xx, etc).

3.3 Component selection

Now that we know all the inputs and outputs into and out of the system. Its time to select our components. (Note: It may not be the best selection, but its what I had lying around)

1. 1x LCD
2. 1x Keypad
3. 1x Buzzer
4. 1x Memory card
5. 1x Keyestudio soil humidity sensor
6. 1x STM32F411CEU6 black pill

4. Design of the System

To implement a **simple** system may be **easy**, but to implement a **complex** system is **difficult** and requires the system to be broken down into smaller pieces called **units**. The units can be tested individually independent from other units. Now that each unit has been thoroughly tested the next phase can commence. It is time to connect these units together, and this process is called **integration**. Each unit may interact with another unit, and this would be the ideal time to test them together. Now that various units has been integrated and thoroughly tested together the next phase can commence. It is time to give meaning to the integrated units to work together to achieve a particular set of goals, and this will be called a **system.**

4.1. Units

All units must be tested separately with the STM32F411CEU6.

4.1.1. STM32F411CEU6	→ Send messages via UART to the PC, and blink an LED.
4.1.2. 16x2 LCD	→ Send a “hello world” message from the STM32 to the LCD
4.1.3. 4x4 KEYPAD	→ Send the keypad value pressed to the PC via UART.
4.1.4. BUZZER	→ Turn the buzzer ON and OFF using a button press.
4.1.5. SOIL HUMIDITY SENSOR	→ Send the soil humidity data to the PC via UART.
4.1.6. 2GB SDCARD	→ Write a “hello world” message to the SDCARD.

4.2. Integration

All units that must communicate together shall do so with the STM32F411CEU6 as the mediator or central controller to facilitate communication between the various units.

4.2.1. STM32F411CEU6 must capture KEYPAD values and display it on the LCD.
4.2.2. STM32F411CEU6 must capture the SOIL humidity and display it on the LCD.
4.2.3. STM32F411CEU6 must capture the SOIL humidity and store it in the SDCARD.
4.2.4. STM32F411CEU6 must capture the SOIL humidity and it its 0%, then trigger the BUZZER.

4.3. System

Now that all the required units can communicate together effectively. It is time to utilize the use cases to implement and achieve the goals that the system must fulfill.

5. System functionality

Threads and Functions

The purpose of ensuring robust functionality by using an Event-Driven Programming paradigm seemed like the ideal design for this application.

Soil moisture sensor thread

The purpose of this thread is to read the soil moisture levels for the plant.

Keypad thread

The purpose of this thread is to read key presses and execute specific functionality.

SDCARD thread

The purpose of this thread is to log all sensor data with timestamps to use for analytical purposes.

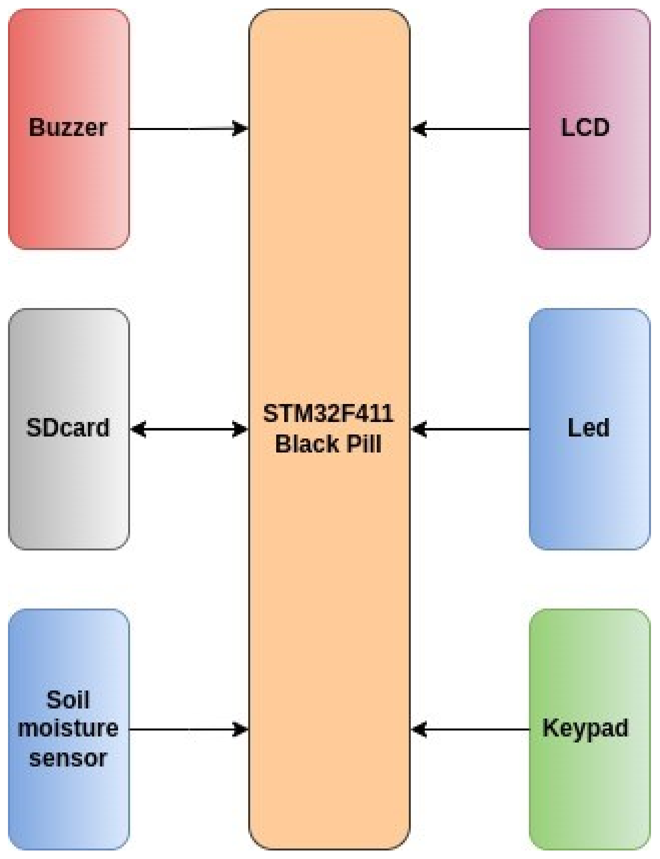
LCD thread

The purpose of this thread is to display messages generated from keypad usage.

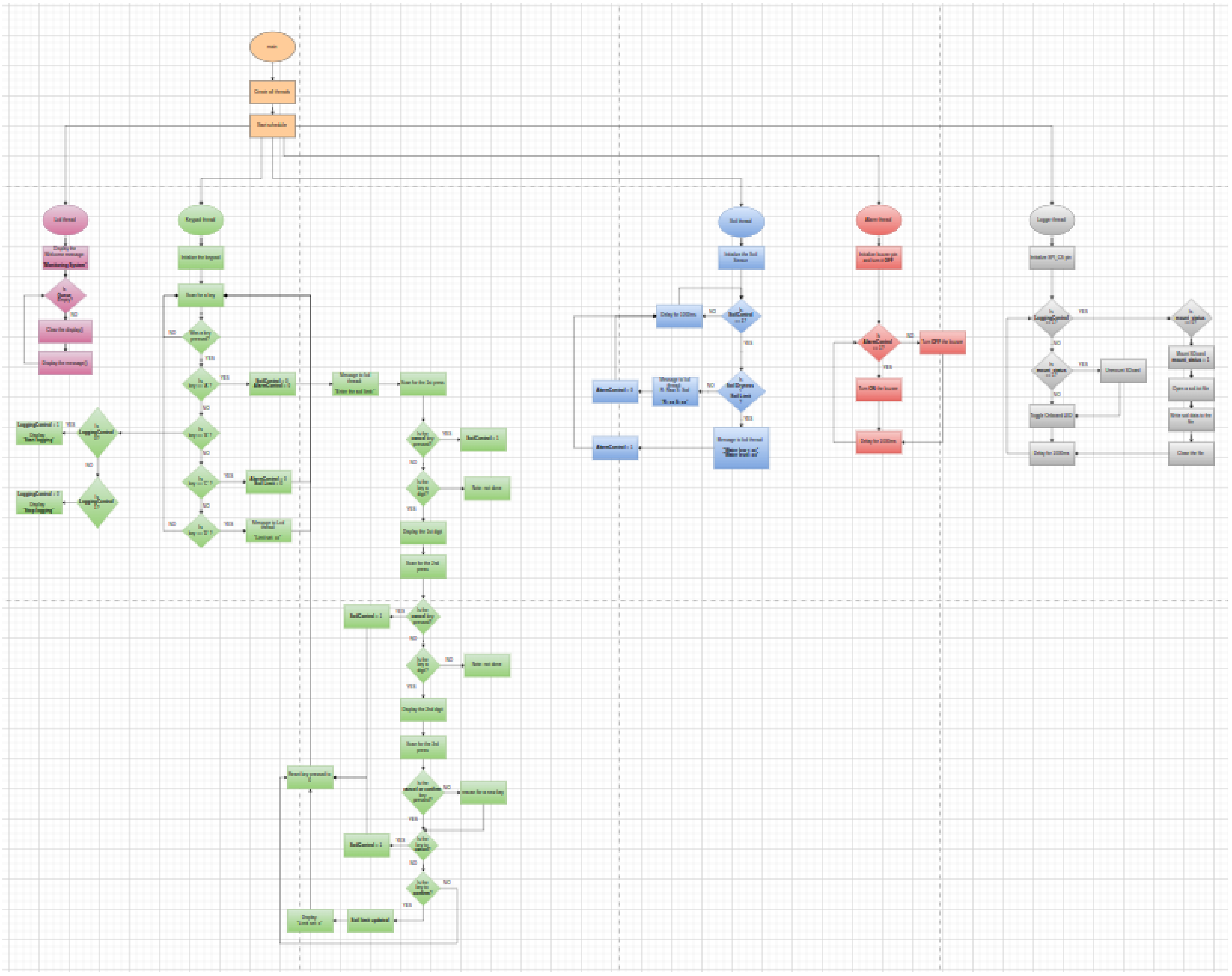
Alarm thread

The purpose of this thread is to notify the user if the soil moisture threshold was breached.

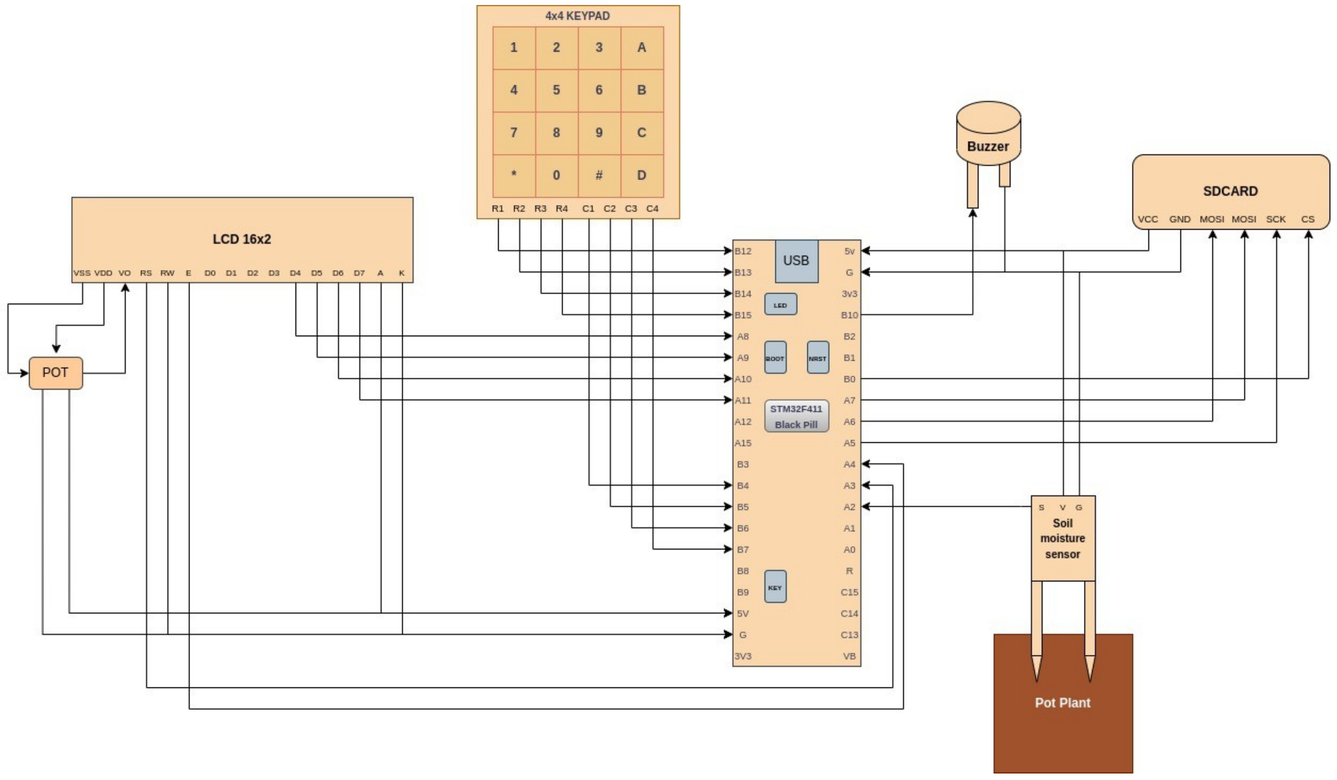
6. Hardware block diagram



7. Software flow diagram of all threads



8. Schematic


4x4 KEYPAD
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
R1 R2 R3 R4 C1 C2 C3 C4
LCD 16x2
VSS VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
POT
Buzzer
SDCARD
VCC GND MOSI MOSI SCK CS
USB
LED
BOOT
NRST
STM32F411
Black Pill
KEY
B12 B13 B14 B15 A8 A9 A10 A11 A12 A15 B3 B4 B5 B6 B7 B8 B9 5V G 3V3
5v G 3v3 B10 B2 B1 B0 A7 A6 A5 A4 A3 A2 A1 A0 R C15 C14 C13 VB
S V G
Soil moisture sensor
Pot Plant

9. Conclusion

Downside of the systems:

The micro-controller in use is overkill for the application. However, the relatively low cost is excellent. The system does not pack much features at the moment.

Improvements for the future:

1. Gsm thread: The purpose of this thread is to send SMS alerts to the user if the water threshold has been triggered.
2. Keypad thread: The user must then be able to add an emergency number. The user must be able to adjust the interval in which data is stored.
3. Logger thread: To add timestamp information to the recorded data.
4. Read the LCD memory to ensure that the correct messages are displayed at all times during operation, and that recovery mechanisms are in place.
5. To add functionality for the system to be Battery-powered.
6. To find a better soil sensor.
7. Finally to design a custom PCB and 3D enclosure.
8. To setup a complete test plan.