WORK PACKAGES:

1. **Continuous Integration:**
   1. Purpose: Allows multiple contributors to update the same repository
   2. Rules:
      1. Directly Pushing to the Master branch is not allowed
      2. Force pushes are not allowed
      3. Pull Request Checklist Template: TO BE DECIDED
      4. Pull Request branches will be based on JIRA Tickets (or some other tool)
         1. Example:
            1. feature/JIRA-TICKET-NO/
            2. bugfix/JIRA-TICKET-NO/
      5. Coding Style: TO BE DECIDED
   3. Features:
      1. Unit Test
         1. Framework: Google Test
      2. Static Code Analysis
         1. Check for MISRA
         2. Tool: TO BE DECIDED
      3. Build
         1. Compiler: TO BE DECIDED
      4. CI pipeline
         1. Deployment: Artifactory?
2. **CMake:**
   1. Purpose:
      1. Can encapsulate unit test, doxygen packages, and install all the remaining dependencies
      2. Can generate build systems for you like Makefiles, Ninja, etc.
      3. Minimizes the platform(Linux,Windows) dependency
   2. Doxygen: Not a must but would be useful to create API Documentation
   3. Build Targets:
      1. Bootloader
      2. Application
      3. Configuration Parameters
      4. All (builds all 3 images), JTAG, or SWD interface is required to flash the combined image
3. **Bootloader:**
   1. Purpose: A custom bootloader would make the system more robust when updating the application
   2. Features:
      1. Ability to update the application
         1. Features:
            1. Downgrade protection
            2. Cybersecurity requirements?
      2. Ability to communicate with the Desktop Application via UART to update:
         1. Application
         2. Configuration Parameters
      3. CRC-32 Check
         1. Purpose: UART uses parity checks. Odd or Even parities can be used to ensure data is transferred correctly. However, the redundancy check fails when 2 or more bit flips occur.
4. **Application:**
   1. A separate application is required to implement the business logic
   2. Modules:
      1. Keypad Controller
         1. Depends on:
            1. GPIO Driver
      2. LED Interface
         1. Depends on:
            1. GPIO Driver
      3. Buzzer Interface
         1. Depends on:
            1. GPIO Driver
            2. PWM Driver
      4. Servo Driver
         1. Depends on:
            1. Timers
            2. GPIO Driver
            3. PWM Driver
      5. UART Interface
         1. Depends on:
            1. GPIO Driver
            2. Timers
            3. UART Driver
         2. Features:
            1. Communicate with the Desktop Application

Diagnostics:

Jump to the Bootloader when a specific command is received (Command Pattern)

Ability to send feedback to the troubleshooters

What is the current version of the application of this unit

* + 1. Flash Interface (Common to UFBL)
       1. Features:
          1. Read and Write Operations

Features:

Synchronous and Asynchronous operations

Image Header extracting

The image header will contain each image’s version number and other parameters. This module needs to search and find (which could be implemented in a lookup table) and return the image header structure to be processed.

* + 1. JSON Parser
       1. Purpose: Serialization of data which minimizes endianness and data type word length when transferring files between different machine architectures.
          1. Features:

Parse JSON data

Send received data to related modules (for example to the Flash interface to save the data in Flash)

1. **GUI Desktop Application:**
   1. Purpose: The product needs to be customizable, and an environment is required to customize the parameters (servo motor angle, number of wrong entries, etc)
   2. GUI Application will be developed in QT Framework in C++
   3. Widget Class will be used to speed up front-end design.
      1. Features:
         1. It is used to configure parameters
         2. Is used to flash relevant images
         3. Ability to detect the USB COM port that our microcontroller is connected to and will initiate a session.
            1. Error Handling when,

The COM port is occupied

The microcontroller cannot be found

* + - 1. Able to use common data transfer protocol to be in sync with the microcontroller
         1. I.e when the data is too large to be transferred (UART 8 bytes), data needs to be split in chunks and the count needs to be in sync
         2. Protocol TO BE DECIDED
      2. Ability to read and extract hex or other file formats
         1. After building each image, their associative image header will be added which will contain CRC-32 value, the GUI is required to send the data so after the microcontroller calculates when the data transfer is over, then the microcontroller can verify that it received the correct image
      3. Front-End
         1. Tex box for adding data
         2. Progress Bar to show the download progress
         3. Transfer data button to transfer config/image