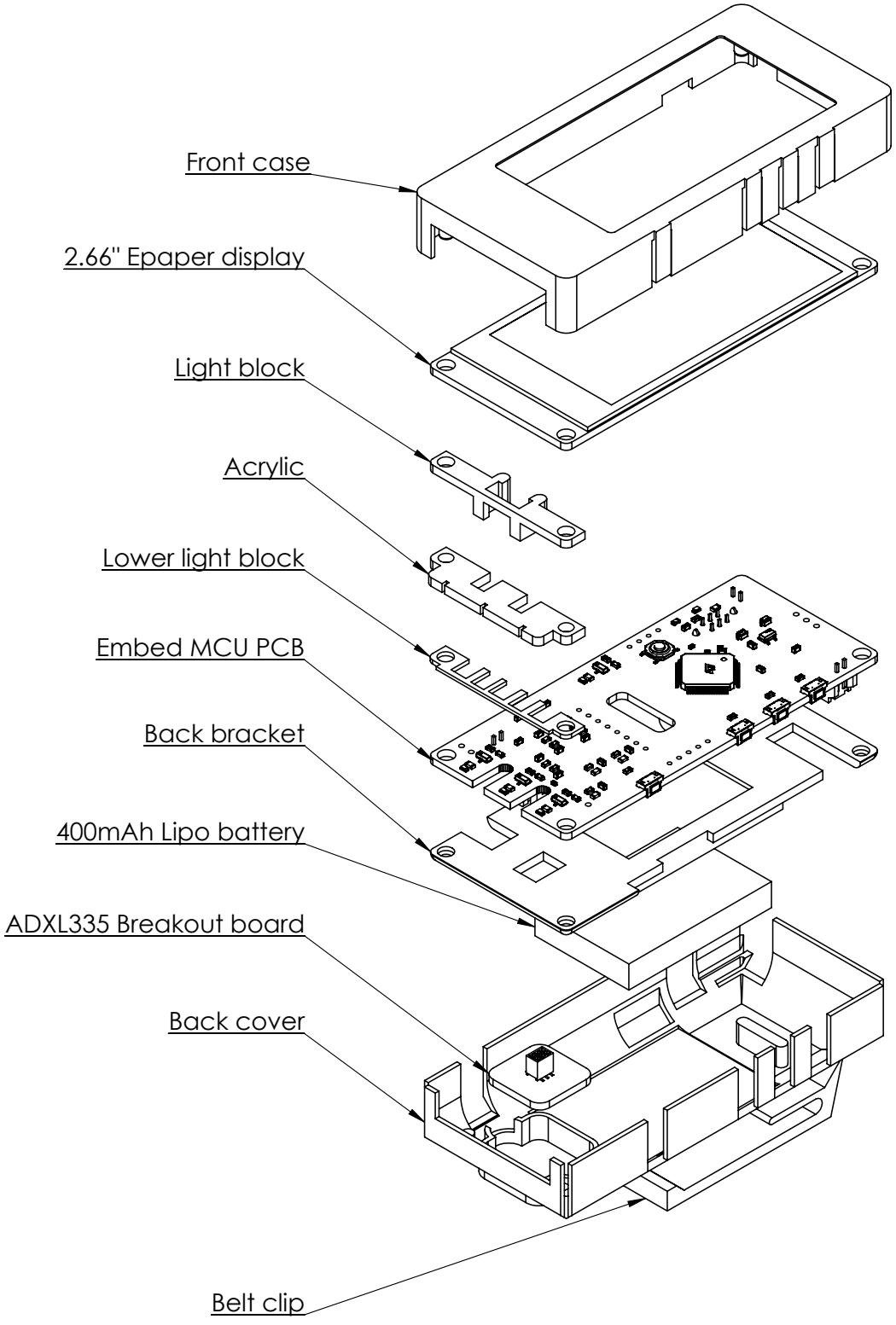
Engineering Design Journal

41070 Embedded Mechatronics Studio 2025



(24428068)

Marcus Frischknecht

× stepper one pedometer project

Table of contents

Personal Profile

Marcus Frischknecht

Portfolio: https://marcusfrisch.netlify.app
LinkedIn: https://www.linkedin.com/in/marcus-frischknecht

4th Year UTS Student studying:
Bachelor of Engineering (Honours) student majoring in Mechatronics.

Figure 1 Self portrait

41070 Embedded Mechatronics Studio is an extremely exciting subject to me and I hope to make the most of every opportunity given! I am a motivated fourth year Mechatronics student and am very passionate in this realm of engineering. This subject is of high interest to me as it provides the tools and opportunities to learn new skills specifically PCB design using industry software 'Altium designer' and access to the signals lab which has high quality tooling including solder rework stations and oscilloscopes. This semester will be the first time I design schematics at the component level and work with SMD components.

Figure 2 Discord camera personal project

My passion for Mechatronics is explored through personal projects such as a compact camera that auto uploads photos to discord. I enjoy learning programming skills, MCAD, and now with this subject – a proper understanding of designing electronic circuits with ECAD paired with new soldering techniques to bring it to life (e.g hotplating). The learning outcomes of this subject will lead me to produce much higher quality, smaller and thinner work aligning much closer to what's produced in industry.

My future career goals include the strong desire to work in the consumer electronics industry. The pedometer project worked on in this subject and the flexibility of the assignment's requirements allow much creativity and the final product to potentially align close to a 'off-the-shelf' consumer product.
UX product design is a small passion of mine and I hope to implement this passion within my work to increase the quality of the project and have a suitable project I can share with design firms on a portfolio.

My goals for this subject include:

- Embedding a MCU for the first time.
- Gaining exposure to the STM32 microprocessor and development environment.
- Utilising the free courses provided to learn Altium Designer
- Developing finer soldering skills & hot plating
- Working with my team to design and produce a deliverable that is innovative and differs from the conventional solutions for this subject.
- Attempting to make my team's project as small and thin as possible within our student capabilities.

My 'highlighted' skills to contribute to my team includes:

- Prior embedded firmware experience. (ESP32, Arduino)
- MCAD SolidWorks experience + 3D printing.
- Prior team leading experience.

My team role and contribution

Team role and workload distribution

This semester, my team has designated me to be the 'team leader' for the project due to my prior personal project and leadership experience. With this responsibility, it's crucial I have a strong vision for the finalised version of the project and ensure I can transfer this vision to team members to ensure everyone understands how their subsystem will contribute to the final project and how to seamlessly integrate it.

Leadership responsibilities not only include technical involvement with the project but also many soft-skills including critical thinking, resilience to set-backs, good communication, time-management, and ability to give constructive feedback to team members.

As with all team members, technical sub-systems and subroutines were allocated to my workload, however I would also like to share, I took on the unspoken 'overall product design' responsibility for this project. This resulted in much collaboration with team members to ensure the work they produced was compatible with and helped achieve the desired features and 'end-goals' of the project.

Two variants of the project

It's important to note, our group produced two variants of the pedometer project.

1) *Embedded MCU PCB a.k.a 'Custom board'* 2) *Shield PCB*

Due to our little experience with STM32, let alone embedding an MCU, we were not confident we could successfully complete the custom board. As a backup we decided we would also design an Arduino Uno compatible shield. The two variants were to be designed simultaneously, although the custom board was always assigned higher priority. With this self-assigned additional work, the responsibilities were increased on group members for separate PCB and MCAD designs. Additionally, the subject coordinator granted our group the exception to have the custom board professionally assembled from our PCB manufacturer if we solder the shield PCB ourselves (Mainly because the MCU footprint is difficult to manually solder). This portfolio with ECAD & MCAD design choices will focus on the custom board variant.

Contribution to the project

Table 1 Personal contribution to the project

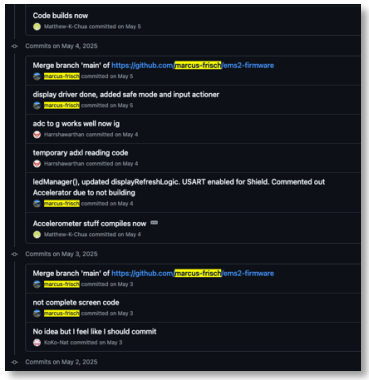
Leadership	Impact
Appropriate delegation of tasks to team members.	Resulted in efficient and timely completion of tasks and not overly burdening members.
Setup of comms and collaborative platforms (Git, CubeIDE shared projects, Powerpoint)	Advanced Git Discord webhook integration kept all team members informed on new firmware commits.
Ensuring each team member understood their responsibilities and project progress.	Little wasted time and effort from group members.
Consulting and giving guidance to team members *E.g Firmware subroutine algorithm ideation, soldering techniques.*	Ensuring team members had a thorough understanding of the task for efficient work and achieved product feature goals.
Purchasing components	Simplified the process of internal peer transactions and management of inventory.
Technical tasks (Discipline specific)	**Impact**
Embedded MCU (main subsystem)	Produced a project closer to an 'off the shelf product'. Aligned the project closer to industry standard/expectations.

Buttons, Accelerometer, Display (peripheral subsystems)	Unconventional 'edge mount buttons' drastically improved UX and product dimensions.
Overall firmware structure	Simplified the process of system integration and made it easy for peers to work within a modular firmware system.
E-paper display driver + Display subroutines	Unconventional E-paper display makes the project stand out and promotes good UX & intuitive GUI.
Code review & debugging	Optimised peer code for improved maintenance.
MCAD design (w/ renders)	Drastic focus on UX with unique features gained favourable attention to the project. Renders increased quality of presentations and tutor engagement.
PCB design, Sub-system placement + custom footprints	Custom footprints provided more freedom for part choices than what Altium MPS could provide.
Achievements	**Impact**
High creativity with original and innovative product design	Created a healthy, motivated team moral, increased peer zeal for the project.
High quality presentation slides	Engaged tutors and increased the teams drive for high quality work in all project aspects.
Established a brand identity tying product design, branding logo, presentation, and documentation deliverables together.	The project felt less like an 'uni group project' and more like working on a real product with consumers in mind. Higher quality output and effort from all team members.
Completed Altium Designer basics course	I was better trained on Altium designer and could perform tasks efficiently.

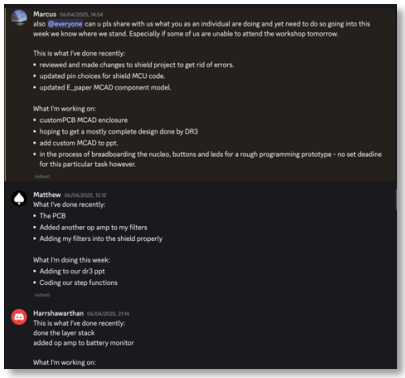
Evidence of my contributions from communication/collaboration tools

Table 2 Evidence of personal collaboration

Altium365 Contributions



GitHub firmware contributions



Online team communication (Highlighted example of leadership)

System block diagram

The annotations indicate the subsystems that I was responsible for in the project.

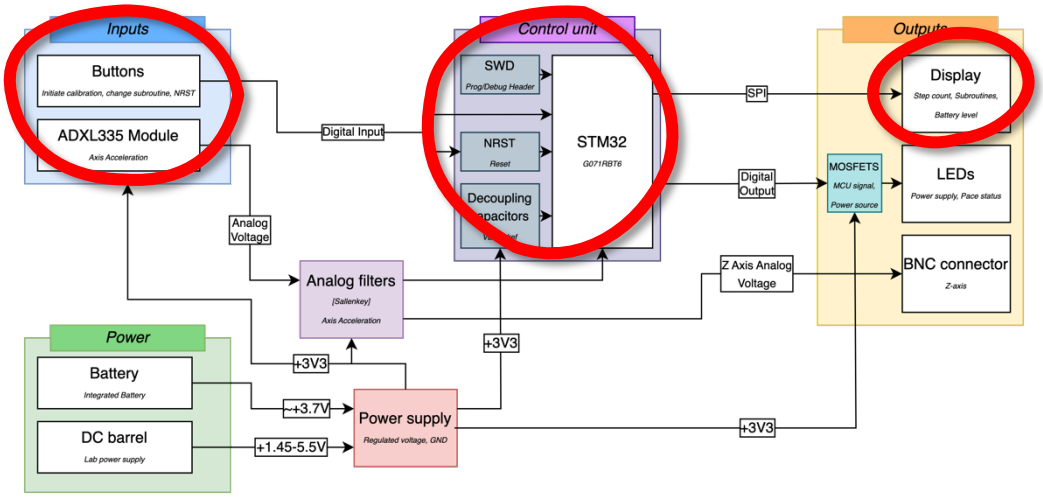


Figure 3 System block diagram

Hardware subcircuit: Embedded MCU & co.

Overview & Specifications

To better align the project with industry, learn a new skill and make the most of the support offered by tutors, I suggested to the group we should attempt embedding the MCU onto the PCB rather than using a development module as suggested by the subject's project brief. Direct benefits of embedding the MCU include a drastic size reduction in the projects overall enclosure and simplified MCAD with no need to worry about header pins passing through the outer-casing.

Figure 4 Embedded MCU

Functionalities of the MCU

- Ability to process Analog voltage output from the ADXL accelerometer and identify 'steps' accurately in any axis amidst other acceleration data output from the accelerometer. Steps detected should be counted.
- Counted steps should be displayed to the user using some sort of graphical display. (I2C or SPI enabled functionality)
- Process additional input from the user such as button presses to invoke different software subroutines (e.g sensor calibration)

Requirements table for MCU choice

Table 3 MCU requirements table

No.	Requirement	Min.	Fav.	Max.	Units	Notes
1	ADC capable GPIO count	4	-	-	-	Required for ADXL and battery voltage monitoring
2	Additional GPIO count	11				Needed for buttons, Display, leds, e.t.c
3	Operating voltage	3	3.3	5	V	Avoid need of Logic level conversion shifters
4	Dimensions	1	1.5	4.94	cm	Balance needed between ease of soldering and minimising PCB sizing requirements
5	Flash memory	32	-	-	KB	Arduino UNO storage flash as minimum
6	Processor speed	16	-	-	MHz	Arduino UNO speed as minimum
7	Peripheral Capable	I2C	SPI		-	Interface for potential display modules
8	USART/Serial Capable	YES				Needed for debugging or unexpected peripherals
9	Compatible HAL Library	YES				Simplify firmware development

10	Solder footprint	Extended legs	Tutors recommended against BGA for ease of soldering and manual correction

Highlighted design choices

The **STM32G071RBT6** was the MCU decided best for our project.

- It met all our MCU requirements stated in the above table.
- We understand STM32 is the market leader for embedded MCUs, so it'd be good to gain familiarity with their ecosystem. (Alternatives included ESP32 & Atmel)
- The G071RB is the same MCU used on our Nucleo boards for 48622 EMS, so we can use the same codebase with our Nucleo's for testing/debugging in case we observe weird behaviour with our embedded board. *(We did, and having the Nucleo gave piece of mind).*

System architecture - Hardware

In my learnings, to successfully embed the MCU, there are three supporting systems required to be implemented on the PCB. Following the MCU datasheet is essential for understanding component requirements.

1. Decoupling power capacitors.
2. Reset circuit (NRST button)
3. Programmer interface (SWD)

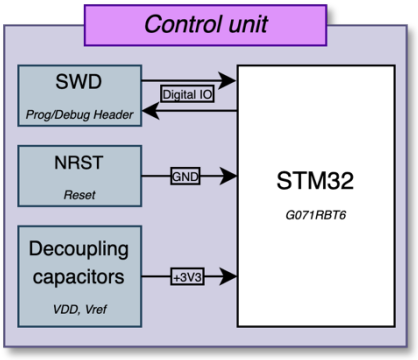


Figure 5 MCU specific block diagram

Decoupling power capacitors

A decoupling capacitor can be considered a 'tiny power reserve' that can provide an almost instantaneous supply of power when required. Decoupling capacitors are required because an MCU's power consumption is expected to vary over the course of its program execution. This is particularly noticeable when the MCU is interacting with peripherals or utilising different internal circuitry such as Analog Digital Conversion. Within our project ADC is used and this results in increased current consumption momentarily when the MCU performs an ADC reading. Due to the combination of increased current and internal resistance along the trace length from the power supply to the MCU, it is expected there could be a voltage drop across the trace momentarily. Any sort of voltage drop (brown out) can lead to improper function of the MCU and so decoupling capacitors are required to supply the necessary current immediately maintaining the ideal voltage level to the MCU. Within the MCU datasheet it is stated the V_{DD} and V_{REF+} pins require decoupling circuitry.

Thorough learning from mistake

A mistake was made in my initial design, resulting in faulty ADC. Learning what went wrong and devising a plan to fix the already produced PCB was a very profitable learning opportunity. See reflection section.

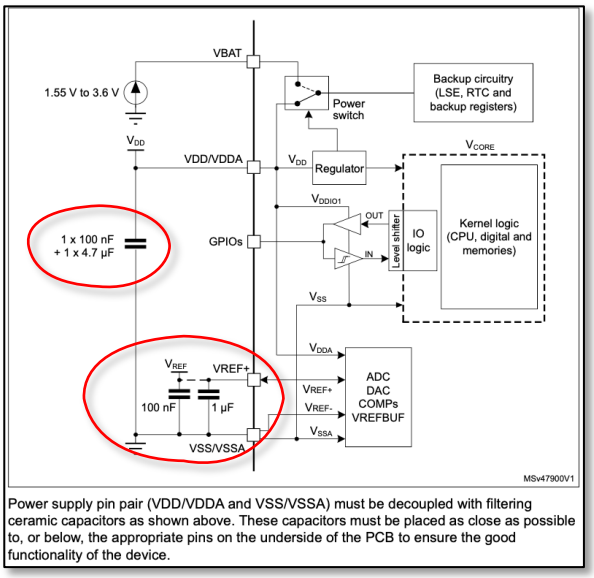


Figure 6 Power characteristics schematic STM32 datasheet

Reset circuit (NRST button)

The MCU offers a physical NRST (reset pin) that can be used to clear registers and restart the MCU. This is useful during firmware development in the event the program crashes or gets stuck in an infinite loop. Our G071RB features an internal pull-up resistor allowing for the physical pin to be left floating, however it is extremely recommended to implement a momentary push button that pulls the pin to GND. By implementing the button, the MCU can be reset without the need to disrupt the power supply.

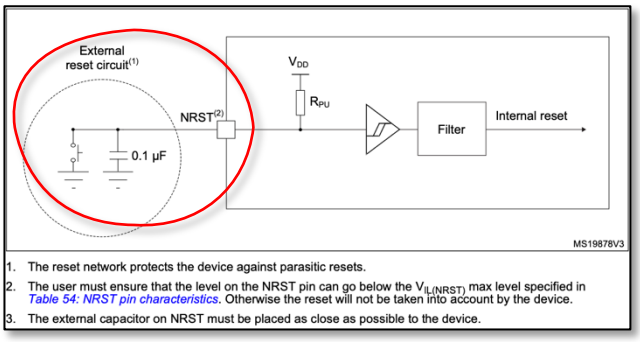


Figure 7 NRST schematic STM32 datasheet

If we desperately needed to save space on the PCB, it is possible we could perform a software induced NRST using the HAL library paired with an appropriately configured external interrupt utilising our existing user-interface buttons. (E.g two buttons pushed at once).

Implementing the pull-down push button suggested in the datasheet was easiest for us and so that's what we did.

Programmer interface (SWD)

An interface is required to allow programming of the MCU and utilise debug functionality. Serial-Wire-Debug is the commonly used interface for ARM based STM32 MCUs compared to alternatives such as JTAG. During design, I recalled our previously mentioned Nucleo development boards include a ST-Link programmer module that can be used to program external MCUs. By using this ST-Link, there's no need to purchase another programmer and we can also use easily use the live debug features taught in 48622 EMS. (Live variable expressions & breakpoints).

The physical interface chosen for our SWD was standard pin headers. The benefits of this decision allow us to use jumper cables to easily connect to the headers on the ST-Link and once all firmware development is complete, we can de-solder the headers to maintain a sleek and slim overall product chassis. Alternatives to headers include Pogo-Pins or a FFC connector, but these are expensive and generally more complicated compared to standard pin headers.

System architecture - Software

Since my focus was the MCU, I assumed position of 'Firmware lead'. I would like to clarify, with this responsibility, I did not write all the software subroutines. Rather, I took the responsibility of initialising the CubeIDE project, configuring peripherals and planned how to shape the overall firmware structure. This included breaking subroutines down into their own functions and how subroutines should handle common peripheral IO without dependence on other subfunctions. (E.g, Code responsible for drawing graphics on our display should be abstracted in its own function and not nested throughout other subroutines.) To achieve thorough subroutine abstraction, the entire firmware revolves around a central global variable named the 'stepper' struct. This struct contains all the variables responsible for the logic of the firmware including variables influencing state machines, peripheral outputs and sensor readings. If one subroutine would like to influence another subroutine, it can simply change the value of a variable within the stepper struct and the other subroutine will respond accordingly to the new value.

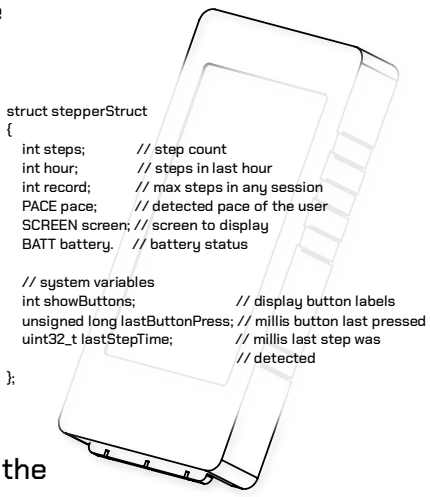


Figure 8 Stepper struct

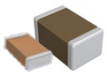
The benefits of abstracting the subroutines include:

- **Allows parallel firmware development**
  Other team members can simultaneously develop their subroutines with minimal dependence on other peers subroutines to be completed first since they can anticipate how the stepper struct will be updated.
- **Easy integration**
  Subroutines can be easily integrated into the whole system or temporarily disabled to assist with individual subroutine/subcircuit testing.
- **Clean code**
  Firmware code is much easier to comprehend and promotes ease of future maintenance.

My dedicated subroutine was the Epaper-display driver code. – Discussed later in this portfolio.

Hardware components

Table 4 MCU hardware components

MCU	Capacitors	Push button	Headers
			
STM32G071RBT6 - Meets our defined requirements	- 1x CAP CER 4.7UF 25V X5R - 1x CAP CER 1UF 16V X5R - 3x CAP CER 0.1UF 25V X5R - Meets datasheet specs	PTS526SK15SMTR2LFS - Slim small push button, does not take up too much surface area	2.54mm header pins - Standard headers, compatible with jumper cables

Calculations & Simulations

The only calculations required for embedding the MCU was finding decoupling capacitors with an appropriate temperature coefficient to meet the 'nominal capacitance' specified within the datasheet. Working with capacitors was a new concept to me this semester. When looking at Digikey, I was overwhelmed by the number of capacitors with the same 'nominal capacitance' and I wasn't sure which one to pick. A tutor taught me about different 'temperature coefficient' classes and provided a graph from *The Art of Electronics: The X chapters* to assist me in my decision.

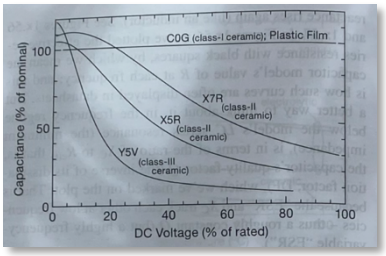


Figure 9 Temperature coefficient chart Art of Electronics the X files

There were not really any electrical simulations I could perform. The closest thing to me performing a simulation was when I initially ran manufacturer display driver code on the Nucleo to test if I could get I2C/SPI libraries working before entirely committing to a STM32 > ESP32 chip. (I had little experience with using external C libraries with STM32.)

ECAD Schematic design

Schematics

When designing electrical schematics, I tried my best to follow industry conventions. This includes all components annotated with values & part numbers, PWR and GND directions, minimal wire crossovers, compartmentalising peripherals and appropriate use of net names and ports. Within the project I also assumed responsibility of designing the Top-Level schematic for our project along with button circuits within our peripherals .SchDoc file.

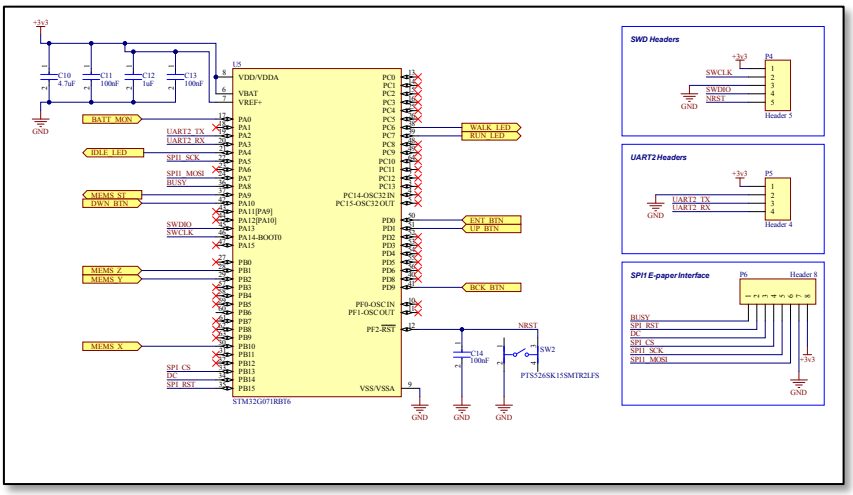


Figure 10 MCU Schematic

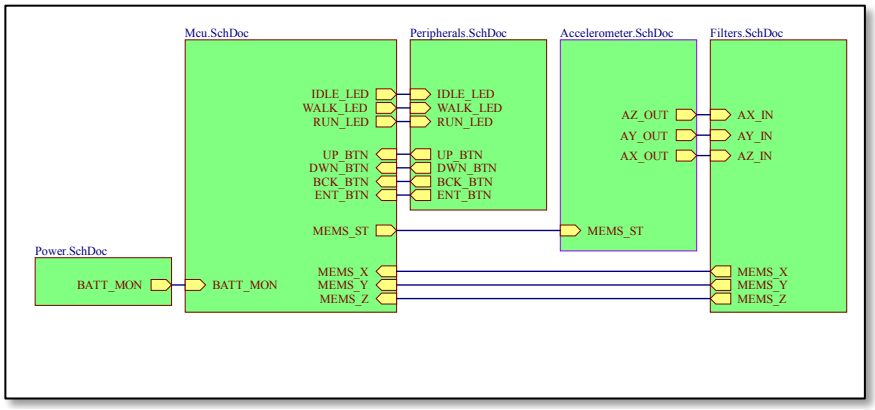


Figure 12 Top level schematic

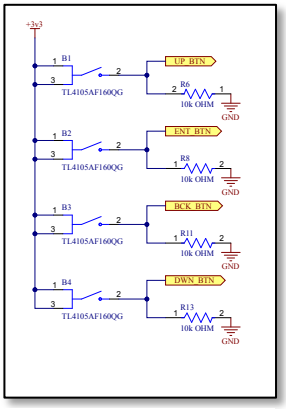


Figure 11 Buttons, Peripherals schematic

Schematic calculations and simulations

As previously discussed within the 'Overview & Specifications' portion of this portfolio, there was no technical calculations or simulations required. Suitable capacitors were chosen based on their voltage and temperature coefficient classes to achieve as close as possible to the ideal nominal voltage specified within the MCU's datasheet.

Pull-down resistors were used for the buttons to prevent floating pins receiving false button presses. 10K Ohm resistance was chosen to limit the current flowing into the MCU and prevent damage occurring.

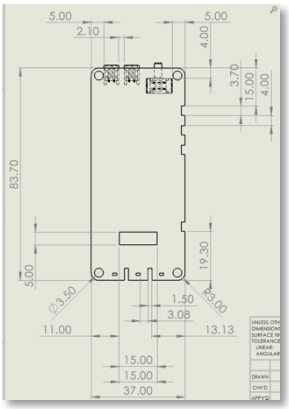


Figure 13 MCAD PCB drawing

PCB design

PCB layout

My responsibilities for PCB design within the project not only included the design of the embedded MCU subcircuit but also the overall PCB shape and placement of other peer's subcircuits. These decisions were allocated to me as I was responsible for the Mechanical-CAD and overall product design of our project.

To begin the design process, I initially created a mock MCAD model in SolidWorks importing STEP files of some of the bigger components planned to be used on our board (JSTs, Switch e.t.c) along with LEDs. Arranging the components around allowed me to experiment with different dimensions and grasp how much surface area our PCB had. Once some initial dimensions were formed, I prepared a Mechanical drawing outlining the PCB edges, and the ECAD PCB design process could begin.

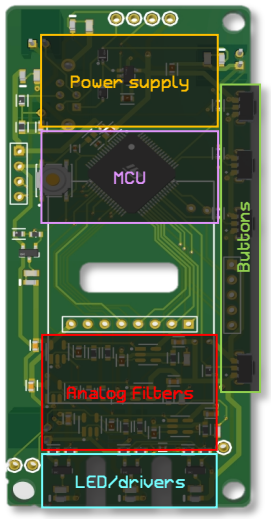


Figure 14 Compartmentalised subsystems

Subcircuit components were grouped together and positioned strategically on the PCB. Each decision heavily considered how it would impact the user experience (e.g intuitive placement of buttons) along with electrical properties *(e.g noise & reducing trace bends for efficient signal flow)*.

Advice from tutors included distancing the analog filters from the power supply to reduce the chance of interference corrupting our sensor readings.

An important design goal for the project was keep overall product dimensions to a minimum. This included careful analysis for determining the optimal component placement stackup to reduce the overall thickness of the product. It was found placing 'taller' components on the PCB's rear side and 'sandwiching' the smaller SMD components between the display module assisted us in designing a thinner product.

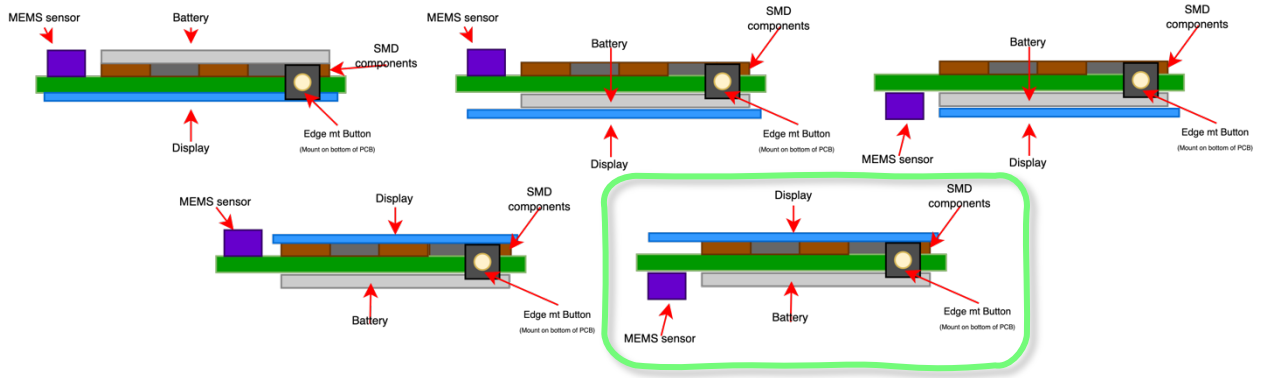


Figure 15 PCB component stackup

MCU Subsystem component placement inc. routing & vias

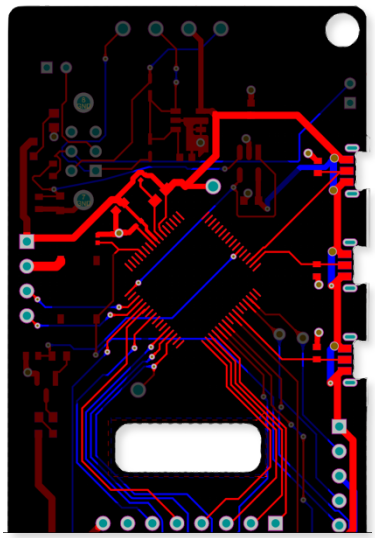


Figure 16 Highlighted MCU traces

The MCU subcircuit was strategically placed in the center of our PCB. This placement eased the routing of all connected peripherals to the MCU and assisted in separating the analog filters from the power supply.

Additionally, the MCU was tilted 45 degrees and GPIO pins were dedicated closest to the peripherals they interfaced with. This helped shorten traces and reduced the number of traces required to run parallel reducing the need to widen our PCB dimensions.

The traces highlighted demonstrate my contributions to PCB routing. In the top left quadrant of the MCU, pads for the decoupling capacitors are visible. In my learnings, it was extremely recommended to place the capacitors as close as possible to the MCU pins. Mindful that I may need to solder these capacitors manually, I did not want to bunch up the 0603 & 0403 package sized capacitors too close together or too close to the very fine MCU pins. *(This was a wise decision as I unexpectedly needed to later rework it manually with a soldering iron).* A possible design alternative was to use a via and place the capacitors on the rear side of the PCB, however since our team was having the board pick-and-placed, it was much cheaper to have all SMD components placed on the top side of the PCB.

To promote efficient signal flow, trace bends were kept to a minimum, set to a 45 degree angle and a GND pour was used to simplify routing and reduce EMI noise.

Design rules

Table 5 Design rules

Rule	Minimum
Clearance	0.11mm
Trace Width	0.17mm
Via Diameter	0.3mm
Via Hole Size	0.2mm
Silk To Silk	0.2mm
Silk To Solder Mask	0.1mm
Solder Mask Sliver	0.1mm
Hole to Hole	0.254mm
Routing Corners	2.54mm
Board Outline	0.2mm

JLCPCB was our chosen PCB manufacturer. To ensure our PCB designs were compliant with their capabilities, our Altium PCB project design rules were configured to match what was published on their website with a little stricter tolerance to ensure we really were submitting a compliant design.

Component and footprint selection

MCU footprint - LQFP

Tutors strongly advised we choose a MCU without a BGA footprint. Our G071RB has a 64LQFP footprint, exposing each pin as an extended leg. This makes it easier for manual reflow work.

Decoupling capacitors - 0603

To keep our PCB small, we ideally wanted the smallest SMD packaged size components while keeping in mind manual soldering feasibility. We were informed the lab we were working in had an abundant supply of 0603 package sized resistors and capacitors which may be useful if we had a calculation wrong and needed to change a components value.
When ordering our assembled PCB from JLC, we discovered they did not have 0603 sized capacitors in-stock with an appropriate temperature coefficient to meet our nominal capacitance. Because of this, two 0402 sized capacitors were used and we hoped we did not need to manually re-solder them.

Buttons – edge mount, right angle

With a hyper-fixation on trying to design the best user-experience for the device, careful consideration went into the selection of an appropriate push button. I realised, by implementing 'right-angle' buttons, not only would the ergonomics of the device be dramatically improved, but it may also provide more surface area on the PCB for other components and routing. During my search of available buttons on Digikey, I stumbled across 'edge mount' buttons which essentially countersink the push button into the PCB – this results in a reduction of the PCB's thickness and when paired with thoughtful MCAD, the 'mechanical push-button' mechanism can be positioned in the center of the device further promoting natural feeling ergonomics.
The component footprint for our buttons includes a board cutout.

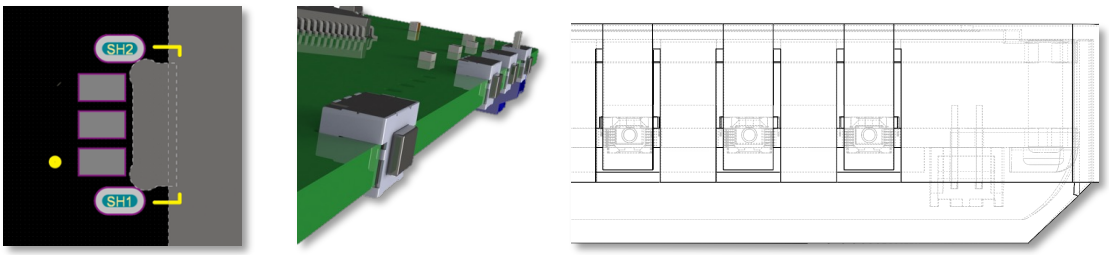


Figure 17 Button footprint, CAD render, CAD hidden lines

SWD, USART, SPI Display – 2.54mm header pads

Standard 2.54mm pitch header pads were used for simplicity and compatible with jumper cables.

Because our PCB and the Display module are the same size and the wires need to be kept short with minimal slack to fit within our small enclosure, I realised soldering may be difficult. This is due to our product assembly requiring our PCB to press tightly against the display driver module. To overcome this challenge critical thinking paired with creativity was used to design a 'wire pass-through' allowing the wires from the display to pass through the PCB and into the rear cavity providing much more room with the soldering iron to solder the display cables.

Figure 18 Cable pass through

Subcircuit testing & Validation

Visual validation

Upon receival of our assembled PCB from JLCPCB, the first thing I did was give the board a thorough visual inspection.

- **Checked component pad population** ✅
  I ensured there were no unexpected unpopulated pads on the PCB and that each populated component visually matched what we expected. One capacitor pad was found un-populated but that was semi-expected due a message we received saying it couldn't be populated due to component shortage. *(This was strange since their inventory webpage listed >16,000 in stock).*

- **Checked all designator component pairs** ✅
  Made sure components were placed in the right orientation. This included the MCU and diodes.
  This check was immediately performed as JLCPCBs populated preview on their website showed the MCU orientated differently to what we designed in Altium. Upon inspection, everything was soldered in its correct orientation.

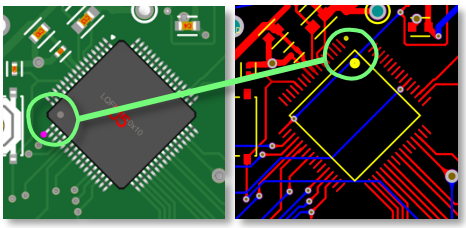


Figure 19 JLCPCB wrong orientation preview

- **Checked solder job with microscope** ✅
  Visual checks was conducted to check if there was any unwanted bridging between components or components lifted from their pads not soldered down correctly. I manually soldered on the power switch, JST connectors, two Schottky diodes and the ADXL interface. The microscope inspections included JLCPCBs soldering and my soldering.

Electrical validation

- **Continuity probing** ✅
  A multimeter continuity test was used to determine the 'open' and 'closed' states of our power switch before the board was wired up to any live power source. Similarly, cont. tests were performed on any questionably soldered components.

- **Software isolated subcircuit testing** ✅
  Temporary code was written to test the subcircuits individually. Having USART functionality integrated on our board made this step easy with outputting appropriate print messages.
  Tests included:

> ⚠️ **Pass now & fail later**
> Our ADC readings initially passed our tests and were working as expected. Strangely over the course of several hours, ADC readings were completely faulty and of zero use to us. This strange behaviour was not a soldering issue but potentially due to capacitance. Details discussed in reflection.

- LEDs individually addressable and working.
- Individual button presses received and reported correctly.
- Display pads correctly outputting to the display module.
- ADC sensor input reflecting accurate readings.

Due to our board being Pick-and-Placed, we couldn't test the subcircuits as they were being assembled. The testing & validation we performed at the time was as thorough as we could *(except for testing our filter output on the test-rig).* Our board passed all our inspections. ✅🎉

Software subroutine: E-paper display driver

Overview & Specifications

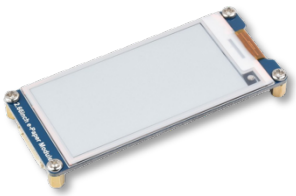


Figure 20 Waveshare 2.66" Epaper display module

Focussing on designing the best user-experience possible, I chose an E-paper display to be the graphical interface to output step statistics and sub-routine information to the user. The large screen resolution caters for a feature rich graphical interface and with the help of bit-map images, can be designed to be extremely intuitive. It's extremely low power, and the zero-light emission *(simply reflects light from surrounding environment)* makes the product look great in virtually any environment *(except dark rooms – but who uses a pedometer in the dark anyway…)*.
It's important to stress, the display is mechanical based *(Ink pigments travelling inside microcapsules)* and so there are certain physical limitations. Extra care needs to be considered when writing code to handle the display for its longevity compared to conventional LCD & OLED panels.

Things that should be considered include:
- Display initialisation *(Partial or Full refresh)?*
- Too many consecutive partial refreshes can damage the display
- Refreshing the display too frequently can damage the display
- Ghosting can easily occur from previous images
- Updating the contents of the display while in motion could potentially damage the display.

The display driver subroutine I wrote is responsible utilises the manufacturer provided SPI library for drawing the graphics on the screen AND controlling the hardware in a safe-manner. My driver is completely independent from any of the other sub-routines. The concept of subroutine code abstraction and it's need within the project is discussed earlier in the MCU part of my portfolio.

The display driver code I wrote is neatly contained within the 'screens.cpp' file keeping our repo tidy and organised.
Only two functions to interact with the display driver are called throughout our entire project.

1. **setupDisp(void)**
   Reserves space in memory for the image buffer and sends SPI commands to wake up the display from it's low power state.

2. **manageDisp(void)**
   This single function simply reads from the stepper struct and draws an image in the buffer with the appropriate graphics and information for the active subroutine. This function is placed once within the loop() of the project (or pre-maturely anywhere else - it does not matter) and will simply send new graphics to the display in a safe manner automatically determining if the display should perform a partial or full refresh. Refreshes only occur when the users pace is detected to be IDLE and if there's a need to

push a new image buffer to the display. *The functions called in the manageDisp() switch-case simply just draw graphics in the image buffer.*

Flowcharts: E-paper display driver

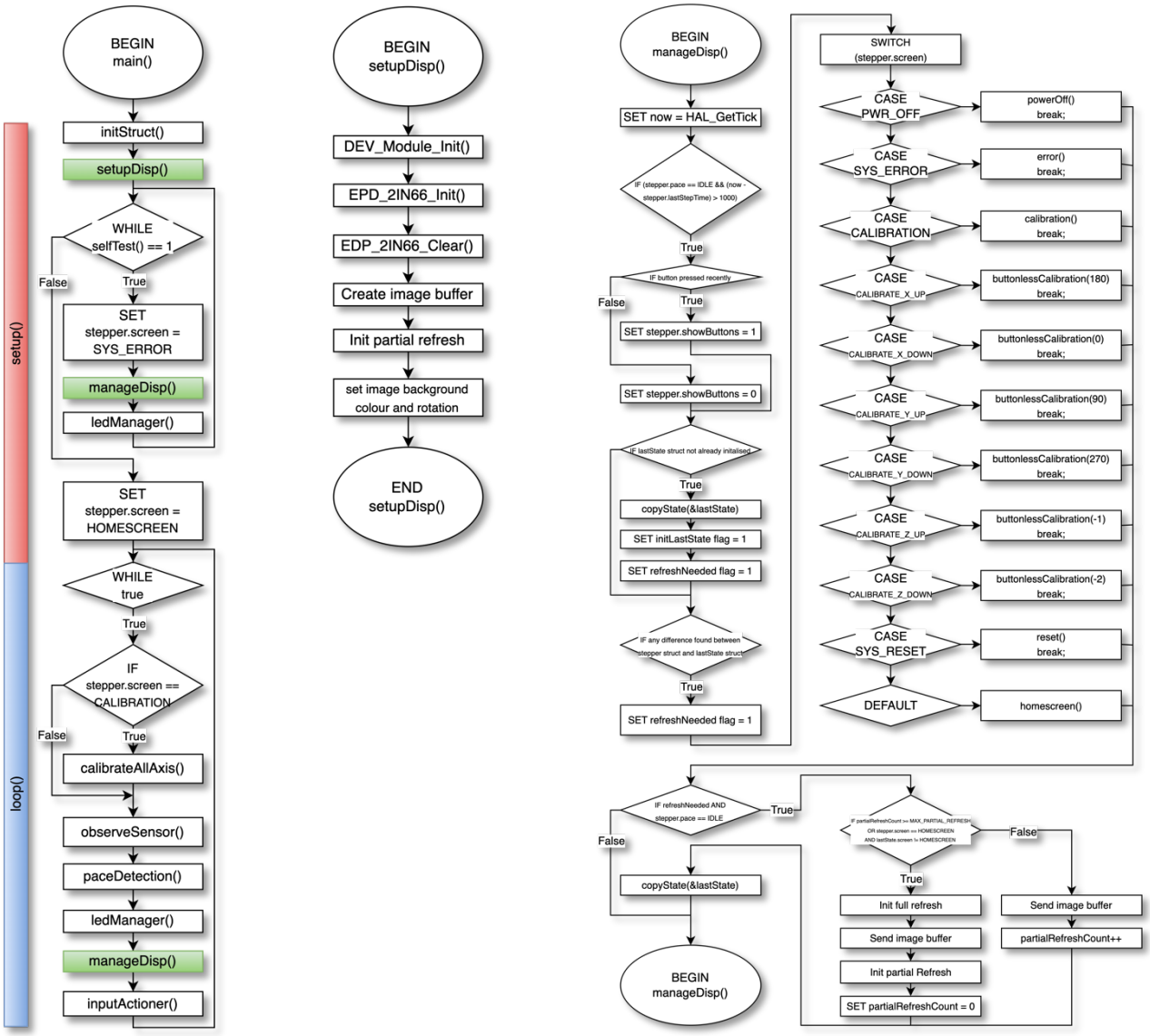


Figure 21 Display driver flowcharts

User guide

- **Bitmap images**

  As previously mentioned, one of the advantages of a large resolution screen is the ability to create intuitive and engaging graphical user interfaces by rendering bitmap images. *(Non-verbal user guidance).*

- **Button labels**

  A unique feature to further enhance the user experience and effortlessly guide the user through sub-routines is the implementation of 'button labels'. This feature conjoins the software interface to the physical world by showing little indicators and labels explaining the buttons functionality within the active subroutine. To reduce visual congestion on the homescreen, if any button has not been pressed for a period of 5 seconds, the button labels disappear. Any button press will invoke the labels to show again. A second press activates the corresponding subroutine.

Figure 22 Stepperone device with button labels

User guide: Story board

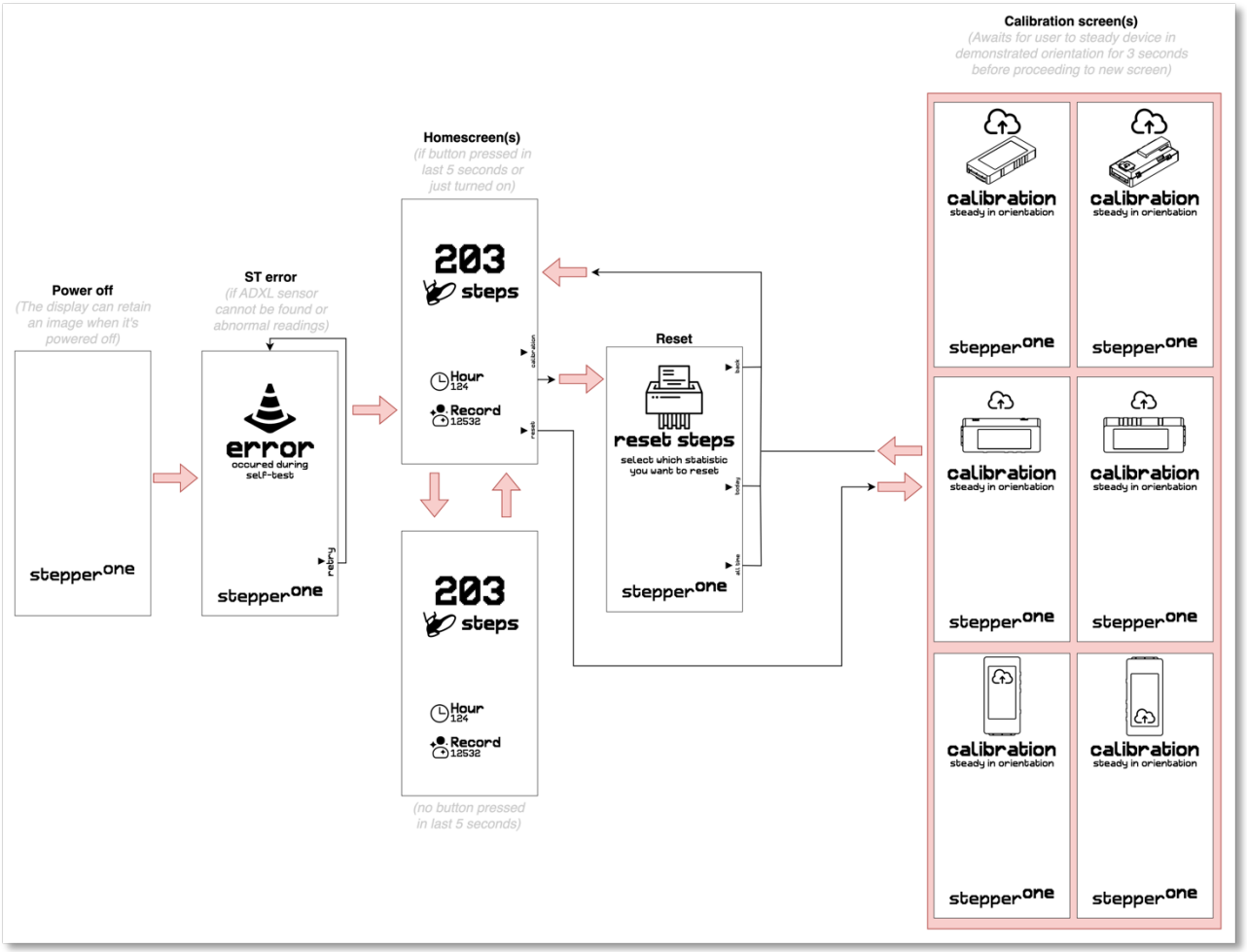


Figure 23 User guide story board

Evaluation, Feedback & Reflection

Group evaluation

By setting ambitious goals, frequent communication between team members, planning for technical setbacks, and continually seeking tutor feedback… I believe our group was able to produce high quality work and each team-member made a valuable contribution, learning much in the process.

Special thanks to Matthew Chua, Nathan Juhandi and Harrsha Gobiram for their efforts within the project. If it wasn't for their active participation and drive to achieve a unanimous vision for the project, the stepperone project would not have been a success.

Based on the tutor marks in last design review and implementing further feedback since then, I believe I can accurately evaluate the group to be within **'Expert'** category for all marking criteria.

Design review feedback

Unmarked design reviews were critical to ensuring our project was making timely progress and our designs were valid. Conscious that tutors need to sit though many presentations for the cohort, we completely re-vamped our PowerPoint slides to make them engaging and better convey the effort the group was putting into the project in a professional manner. As a result our mode grade from tutors increased to 'Expert' from 'Developing'.

Voluntary requests for **feedback on Altium365** allowed the group to receive extra technical feedback in between design review presentations. We made use of this opportunity to better our schematic and PCB designs.

Peer feedback

Throughout the semester, team members and I gave each other feedback, questioning each other's design decisions and providing advice on how functionality or quality of work could be improved. An example of notable feedback given to a team member was the suggestion of 'Buttonless' calibration. This new feature improved the quality of the project and stands out compared to manual button pushes following the subject convention.

Figure 24 Professional powerpoint slide. Simple, clean render

Self evaluation

I absolutely thrived in this subject and tried to make the most of every opportunity I could for myself and the team to learn. I believe I fulfilled the duties of a responsible team leader by encouraging each member at a personal level, promoting a safe space for asking questions and planning ahead for unexpected technical setbacks. Below is my SLOs reflection.

Table 6 SLO Reflection table

Subject Learning Objective	Reflection
Identify environmental and economic impacts of embedded mechatronic solutions in each context. (B.1)	Sourcing (many) components and from multiple vendors taught me projects such as ours do have an environmental impact (packaging, shipping) and can easily add up in cost – even when components by themselves may seem to cost a negligible amount.
Apply systematic electronic and mechanical computer aided engineering synthesis and design processes in combination with embedded software to develop a mechatronic system. (C.1)	I thoroughly enjoyed this aspect of the project. By tightly integrating the firmware development, ECAD and MCAD together within our ideation and workflow, the group produced a product with unique, ergonomic features truly enhancing the user experience.
Use technical skills to develop, model, simulate, prototype, and evaluate electronic circuit designs and embedded software. (D.1.)	This subject exposed me to Altium designer and I enjoyed learning many of its features. Undertaking Altium designer courses throughout this semester was extremely profitable for my learning and saved me time.
Demonstrate effective collaboration, communication, and documentation skills as a member or team leader. (E.1)	I believe I successfully fulfilled my responsibility as team-leader to the group.
Use feedback to reflect for performance evaluation and iterate on a given problem. (F.1)	I will share with you the story how feedback led me to iterate on a given problem that ultimately rescued the project. (The feedback rescued, not me). Please read the reflection below this table.

ADC conversion issues – feedback saving the project

Context:

As hinted at throughout the document, our MCU experienced faulty ADC readings for a substantial portion of the assignment. With faulty readings, this ultimately results in our pedometer being useless; unable to count steps reliably.

Troubleshooting & Debug methods:
- Multimeter probing [pass]
- Reconfiguring software hardware ADC initialisation [pass]
- Trying a different ADXL MEMS Module [pass]
- External Nucleo (same MCU) wire tapped into our PCB vias to report ADC readings [pass]

In our case, when the MCU attempted to read from our analog mems accelerometer, the readings would only read a consistent maximum voltage rendering the system completely unable to detect any motion let-alone foot-steps from the user. Several methods of debugging and technical troubleshooting revealed the the MCU was 'hallucinating' the high ADC values and not reading the sensor correctly. The sensor and analog filter circuitry was operating as expected.

The strangest part of this phenomena was our MCU did initially read the ADC values correctly upon receival from the factory. For multiple hours, firmware development focussed on ADC readings and there was no problem. Only until the following day, we discovered the algorithms we tuned for step detection started to become unreliable. Using 'Live-variable Expressions' revealed the ADC readings were continuously climbing to new thresholds. This behaviour was unexpected. Climbing at an exponential rate, the readings eventually reported maximum voltage and restarting the device had no effect.

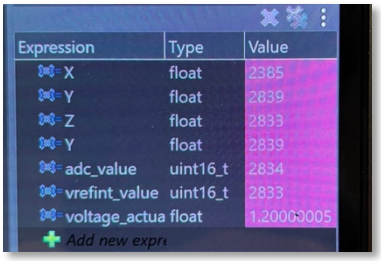


Figure 25 Climbing ADC values Live variable expressions

Thought to be the problem:

My team members and I were puzzled at this observed behaviour. We falsely diagnosed the issue to be thought that an 'electric static' shock damaged the ADC conversion unit within the MCU. We thought this may have been the issue as the board was frequently handed between group members and packed/unpacked on various surfaces. At the same time our USART→TTL adapter broke and it was thought the two issues may have been a correlated electrical issue.

We had a second assembled PCB board spare, but due to financial cost and uncertainty of potentially damaging the second board with static, the group made the decision to not touch the second board until much closer to the final presentation.

We shared our experience with tutors. No insight could be given, the theory of static being the problem was upheld and the encouragement to keep the second board safe was embraced.

All aspects of the first board were operational with an exception to ADC. Due to this, the group continued firmware development and progress was not hindered.

Iteration #2:

The night before the demonstration we unpacked and flashed our second board. We expected it to be fault free like the first board for the first few hours of its life. Within two minutes, our second board had rising ADC values, climbing at a rate much faster than the first board.

Attempting to recover the project's dire situation, within early hours of the morning demonstration was due, a plan was derived. Other group members would consult online resources to investigate if there was a lower lying software configuration issue with ADC. Simultaneously, I was preparing an ESP32 C3 Mini dev board to be tapped into the vias on our PCB board, read ADC values and forward them to the Faulty Embedded STM32 on our PCB using USART.

Shortly into the iteration, the other members found recalibrating V_{REF} frequently on the STM32 drastically reduced the symptoms of rising ADC values. This *(temporary)* software patch seemed

reliable enough to cancel the ESP32 iteration of the project and the groups focus was diverted to other aspects of the project that needed attention.

The feedback:

Later that morning, testing the device for demonstration, we discovered our V_{REF} software patch had failed, and we were back to reading maximum voltage. With no time to implement iteration #2, we accepted our fate, prepared the presentation slides, and tried to tell our sad story in the most engaging and insightful way.

The faulty schematic:
- V_{REF} did not have a direct line to V_{DD}
- The decoupling capacitors were not tied to GND but to V_{DD} instead.

Shortly after our presentation, a tutor came out to meet our group with our MCU schematics on their laptop and pointed our attention to the V_{REF} decoupling capacitors. It turns out I made a mistake and did not follow the datasheet properly. This mistake slipped through the tutor design reviews too.

With this breakthrough, we informed the subject coordinator about our discovery and were graciously given a short extension to sort out the issue. Time for iteration #3.

Iteration #3

Quick critical thinking was required to find a solution to the problem. We needed to reroute power to the VREF MCU pin and ensure the pin was decoupled. The quickest way to do this was to:
- Remove both incorrectly routed capacitors
- Bridge the 0402 pads (establishing a line for VDD to meet VREF)
- Stand a 0603 0.1uF capacitor vertically on an existing pad and use a thin resistor leg to bridge the floating capacitor end to a GND via.

Upon power up, our ADC values were reading as expected and the group could proceed with functional step-tracking.

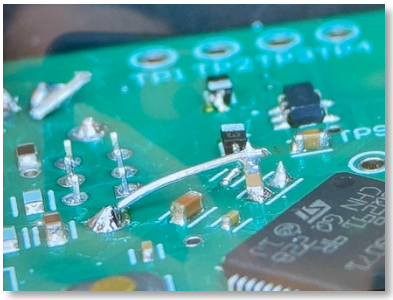


Figure 26 Vertical capacitor fixing VREF

Lessons learnt

This experience has taught me many valuable lessons including

1. Check the datasheet multiple times. Particularly check before ordering the PCB.
2. Pursue feedback from (many) sources. Perhaps 2 sources is not enough.
3. All because the hardware worked in the beginning, does not mean the hardware is correct. It's not clear why it worked for multiple hours compared to minutes on the second board. If it's to do with capacitors charging, I thought that would've been done within seconds. Regardless it was a miracle though because we had other software challenges at that time and debugging faulty ADC readings would've made it much more complicated.

Challenges

A challenge I experienced included overly assigning myself a large workload. With confidence in my prior experience and mindful that I was studying a reduced workload this semester compared to my team members – I assigned myself a greater number of responsibilities unless peers volunteered. At times, the workload was not sustainable for the project and I needed to transfer the responsibility to another member. Next time it may be better to keep some tasks 'floating' and not assign them early to any member. I also need to humble myself.

Strengths

I consider myself a creative person and thoroughly enjoyed considering the user experience and brainstorming ideas to make the best product we could for this project. I hope in the future

there will be more opportunities for me to work on projects with a combined responcibility of Embedded Mechatronic work and Product design. I think I would like to pursue a career in an electronics product design firm. Another strength demonstarted is rapid prototyping and quick critical thinking – helpful for resolving unexpected technical issues quickly.

Referencing

Use of GenAI

GenAI was used in this assignment as a tool to answer my questions, seek clarification on technical concepts and provide some guidance when tutors were not around. I found it useful for:

- Seeking clarification on STM32 MCU capabilities
- Setting up a STM32 CubeIDE project efficiently
- Troubleshooting code
- Asking for help with Altium designer program features
- Asking questions about datasheet content
- Automating code generation
- Seeking electronic concept clarification
- Generating APA references

Links to my chatGPT conversations:
RC filter, STM32 ADC & VREF, STM32 EEPROM, STM32G071RB project help, Altium designer q's

APA 7th References

Analog Devices. (2021, October). *ADXL335: Small, low power, 3-axis ±3 g accelerometer* [Data sheet]. https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf

Horowitz, P., & Hill, W. (2020). *The art of electronics: The X chapters*. Cambridge University Press.

STMicroelectronics. (2022, May). *STM32G071x8 STM32G071xB advanced ARM®-based 32-bit MCUs* (Rev. 1) [Data sheet]. Digi-Key. https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/726/STM32G071_Rev1_DS.pdf

STMicroelectronics. (n.d.). *Getting started with GPIO* [Web page]. STM32 MCU Wiki. https://wiki.st.com/stm32mcu/wiki/Getting_started_with_GPIO

STMicroelectronics. (n.d.). *Getting started with SPI* [Web page]. STM32 MCU Wiki. https://wiki.st.com/stm32mcu/wiki/Getting_started_with_SPI

Waveshare. (n.d.). *2.66inch e-Paper HAT (B): 296×152 resolution, SPI interface* [Technical specification]. https://files.waveshare.com/upload/d/dc/2.66inch-e-paper-specification.pdf

Icons (Creative Commons Attribution 3.0)

Creative Stall. (n.d.). *Shredder* [Icon]. The Noun Project. https://thenounproject.com/icon/shredder-6242964/

Ilham Fitrotul Hayat. (n.d.). *Cloud Upload* [Icon]. The Noun Project. https://thenounproject.com/browse/icons/term/cloud-upload/

Luis Prado. (n.d.). *Running* [Icon]. The Noun Project. https://thenounproject.com/icon/running-6899623/

muse mango. (n.d.). *Traffic Cone* [Icon]. The Noun Project. https://thenounproject.com/icon/traffic-cone-7666943/

reyo. (n.d.). *Achievement* [Icon]. The Noun Project. https://thenounproject.com/icon/achievement-7292127/

Sandor. (n.d.). *Table* [Icon]. The Noun Project. https://thenounproject.com/icon/table-166058/

Srifa. (n.d.). *Step* [Icon]. The Noun Project. https://thenounproject.com/icon/step-1316683/
Agus Purwanto. (n.d.). *Clock* [Icon]. The Noun Project. https://thenounproject.com/icon/clock-746591/

Thank you Michael for the best subject in the Mechatronics course ❤️