PROJ200-P1-2023-24 Group Project

Prepared by: Nicholas Outram

Revision History

Revision	Changes	Date
0.1	Initial version presented to colleagues	Sept 2023
0.2	Draft released to students	16th Oct 2023

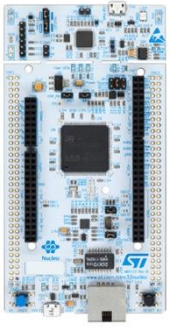
1 SUMMARY

1.1 Introduction

This module has two assessed components: **Individual coursework tasks** (C1W1+C1W2) and a **Group practice element** (P1). Group management is a requirement of this module. The rationale for this is to teach some basic management practice in a relevant context, and to prepare students for their final year project.

1.2 Goals

Greater opportunity for students to work collaboratively; upgrade repertoire / capability of final stage ELEC and ROBOTICS projects; further enhance student employability; students to build the confidence and capability to correctly use modern technology in their final-year projects, ensure add students are familiar with sampling of real time analogue signals in real time, PCB design and connector crimping; use of modern digital interfaces; managing real-time problems in software and hardware; programming and interfacing to a FPGA;



2 OVERVIEW

The theme of this coursework is health. The students are to complete and build a heart rate monitor using a pulse oximetry sensor for input. The analogue signal is picked up using a pulse oximetry probe, which measures reflected light from the blood capillaries in the fingertip. As there is no electrical contact with the body, this is conveniently isolated and safe. The electrical signal is amplified and filtered, before being connected to the ADC on the FPGA board.

The **FPGA is connected to the ADC via a SPI** interface, where the **FPGA acts as an SPI master**. The task of the FPGA incudes reading the samples at a fixed and determined rate and storing them in an internal buffer, all in real-time. Once stored, these samples are available to be read by the MCU when ready. The FPGA shall use its own high-speed clock to perform the sampling.

The **MCU is interfaced to the FPGA using SPI**, where the **MCU is now the master device.** The MCU is able to request and read samples from the FPGA buffer. These samples are then analysed, and heart rate is calculated by the MCU software. This can be logged to the terminal for diagnostic and debug purposes.

An **LCD screen is connected directly to the FPGA GPIO pins** (making note to use the correct logic levels) and is used to display text, including the heart-rate. The MCU can write data to LCD via the FPGA over another SPI interface. Again, the MCU is a SPI master, and the FPGA is a slave device. As characters are written to the FPGA, these samples are buffered by the FPGA and dispatched serially on the LCD screen in real time, respecting the timing requirements of the LCD hardware interface.

Note that from the above, the **FPGA will have three SPI components**: an SPI master (ADC), and two slaves (samples and LCD). All these “devices” are independent parallel hardware.

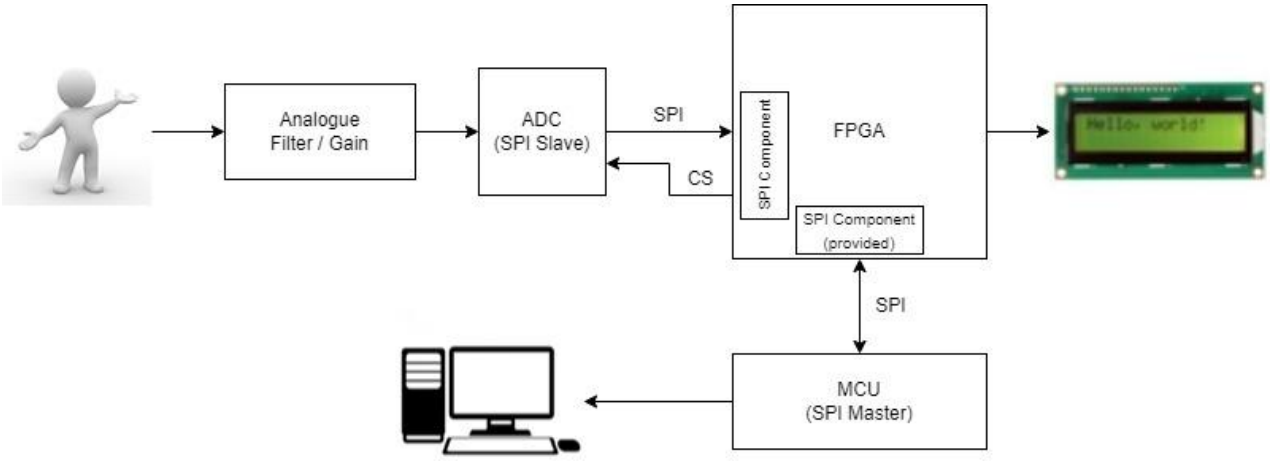


Figure 1. Block diagram for the ECG monitor.

2.1 Input Signal and Signal Conditioning

The pulse oximeter signal is assumed to have bandwidth of 0.01 – 15Hz. The signal is small so needs to be amplified. We must assume that noise is present, so the signal has to be filtered before it can be sampled.

The Gain + Antialiasing Filter block in Figure 1 needs to ensure the signal is large enough to be sampled. It must not be so large that it hits the power rails (saturation), but equally, if it is too small it cannot be sampled (digitised) accurately. Furthermore, we need to remove noise from the signal. The highest frequency of interest is 15Hz, so the filter needs to remove energy about that frequency as much as possible.

The analogue design is partially completed. You are required to modify it to meet your chosen specification.

The analogue system needs to be simulated and tested on prototyping board before being assembled on a PCB and validated. Once constructed, the analogue board shall be connected to the FPGA using a crimped connector.

2.2 Sampling

The signal will be sampled by the 12-bit Analogue to Digital Converter (ADC) on the FPGA board. This needs to be at a fixed and rate of your choice. Note that sampling rate will impact on the effectiveness of the filters to avoid "aliasing". He principle is to sample the signal has fast as possible. This is known as "over-sampling", and (as will be explained) is used to improve the effectiveness of the anti-aliasing filters. However, higher sampling rates require more storage and processing, so you will need to consider this trade off.

The FPGA components you develop to do this need to be verified (showing them to functionally correct) and validated. This includes showing that the sampling rate is the expected value and consistent.

2.3 Real-Time Functions

The FPGA shall handle all hard-real-time tasks, allowing the MCU timing slack to help keep the CPU requirements as simple as possible. This includes both sampling the analogue signal and controlling the LCD.

The data sampled by the FPGA needs to be buffered internally until the MCU is ready to read it. Note that if the internal storage becomes full, this represents a critical error.

The display data sent from the MCU to the display (via the FPGA) should also be buffered internally and sent to the LCD with the correct timing and at a suitable rate. Again, a full buffer represents an error.

2.4 MCU Communication

The MCU communicates with the FPGA via the SPI interface (provided). Example code is provided to demonstrate basic communication. You will need to design and implement some form of protocol to perform the following functions:

- Read the internal samples
- Send ASCII characters to the LCD display

You can either send "commands" as part of the data, or replicate the SPI hardware and have two separate SPI interfaces. The former method is less wasteful of hardware and attracts more marks.

2.5 Heart Rate Calculation

The MCU shall communicate with the FPGA to acquire samples and calculate the heart rate. Techniques include locating the wave peaks or zero-crossings, from which you can estimate the Heart Rate (HR). Students are advised to log samples to the host PC via the serial (over USB) connection so they can develop & test their algorithm offline.

2.6 Display

The heart rate needs to be displayed on the LCD display and updated regularly. How often it is updated if for you to determine.

The display is connected to the FPGA using a parallel interface. It can display 2 rows of characters. The interface allows both commands and data to be sent to the display. Commands include clearing the screen and setting the cursor position. Data is typically ASCII. The display must also be initialised correctly when it is first powered on.

A data sheet and accompanying documentation are provided. These will detail both the communication protocol and the timing. These display devices run relatively slowly, so the FPGA must ensure the interface using the correct logic and timing.

To help the CPU run smoothly you should include “buffering” in the FPGA (typically using the provided SCFIFO component). This “First In First Out” (FIFO) buffer allows the MCU to quickly write a block of commands and character data to the FPGA. The FPGA will then take that data and send it to the LCD display at the appropriate speed. This means the MCU can perform its write operations quickly and move onto the next task, without having to wait for the LCD.

Requirements

For the technical side of the project, you are marked against several requirements, each of which should be **evidenced** in your **final report and presentation**. The marks and criteria for each requirement are summarized below.

Req.	Description	Marks
1	Analogue Circuit Design & Test.	10%
	Update the provided schematic to provide suitable gain and filtering so the signal can be accurately sampled and analysed. Include the following information: • Choose a suitable specification (gain and filter edge frequencies). Provide justification and show component calculations. • Show simulation results (including a Bode Plot) confirming your design • Describe your test methodology (what did you test and how) • Provide test results of the final circuit Remember to consider both time (gain, saturation, signal-to-noise, signal distortion) and frequency domains (bode plot). Your choice of specification will impact on signal amplitude, the sampling rate, noise and accuracy.	
2	PCB Design and Physical Build	10%
	Marks are awarded for the quality of your PCB design. Criteria include component placement, track angles, track size and the use of ground planes. Marks are also awarded for well-constructed (and tested) cabling. The PCB will be 4 x 2 inches or 101.6 x 50.8 mm Size. PCB Must Have Group Identification on the TOP silk layer.	
3	Signal Sampling (FPGA)	10%

	The FPGA shall sample the signal using the on-board ADC. Show evidence that the signal is sampled at the correct and consistent rate, stored internally and that data integrity is maintained. Using the FPGA clock and components written in SystemVerilog, acquire samples at regular intervals and store them in an internal FIFO buffer (use the Intel SCFIFO component). You need to evidence that the signal is sampled evenly, at the correct rate, and that no samples are dropped or corrupted. Develop and describe your methodology for both **verification** and **validation**. For verification, you should write testbenches for each of your components and present the results. For validation, consider using a signal generator and SignalTap (or another method of your own design).	
4	Real-Time Software	10%
	Data from the FPGA shall be transferred to the MCU for analysis using the provided SPI interface. The timing of this needs to be synchronised such that no data is lost and risk of metastability is minimised. You should aim to evenly spread the CPU loading over time. You will need to describe your software technique to achieve this. You also need to present both methodology and test results to evidence correct functionality.	
5	Display	10%
	The MCU shall be able to display the calculated heart-rate on the LCD screen connected to the FPGA. The FPGA shall perform all the timing and required control sequences, such as initialisation, positioning of the cursor, writing characters and clearing the screen. Use the SCFIFO	

	to buffer the incoming characters from the MCU. You should make this as efficient as possible. You should verify every FPGA component using a suitable testbench You should validate the system, describing your methodology and test results.	
6	Heart Rate Calculation	10%
	The MCU software shall perform the heart-rate calculation in software. This should be based on the data acquired from the FPGA. *You are advised to capture real signals on your PC using the hardware (e.g. via a serial terminal). You can then use tools such as Visual Studio or Excel to visualise your signals and develop a suitable algorithm.* Try to make your algorithm as robust as possible (consider effects of noise and sampling). You should then implement your algorithm in your MCU and test / validate. Report on your approach, test methodology and results.	
7	Efficacy	10%
	Provide information on the accuracy and reliability of the final system. This should include including measurement errors due to sampling and quantisation. Comment on the impact of noise and distortion. Critically report on the limitations of your validation technique.	
8	Software	5%

Software and VHDL shall be formatted clearly, paying close attention to indentation. It shall also be appropriately commented and structured. C functions should be full commented, including information about parameters and return types. Code should be split into separate C files. Variable names should be meaningful. HDL should be commented.

At the top of every source file (HDL or C), there should be a block of comments listing each version and developer name(s).

3.1 General Requirements

For full marks, you must do the following

- All FPGA components MUST be written in SystemVerilog and should have an accompanying test-bench. MCU software must be written in C/C++ with the Mbed OS framework.
- A demo that appears to work in NOT enough. For each requirement, evidence of correct behaviour **must** be provided. This should include evidence from demonstration, signal-tap and simulation as appropriate.
- All sources (HDL and C) should be properly formatted and commented and build without modification.
- All HDL modules should have an accompanying test-bench that provides sufficient coverage and is clearly commented. It should compile and simulate without modification.
- You may be asked detailed technical questions about any part of your system. Your responses will impact on the mark awarded.
- Pay particular attention to verification and validation. You will be expected to explain your **methodology** and any limitations.

Group Management

Group work is a learning outcome of this module and is assessed. You are assessed on how each group manages itself.

All group activities are to be recorded in shared OneNote and Planner documents. These (online) documents will be provided to each group by the module leader. Do not create your own.

Formal Group Meetings:

There should be **at least** one formal group meeting each week. For each group meeting, the following minimum information must be recorded in the OneNote document:

- Who is present, and who is not (with any apologies given)
- Numbered items from last meeting
- Summary report from everyone mapped against the assigned actions
- Any Discussion points. Consider the actions set in the previous week and reflect on how viable they were. Was it too much to ask, or maybe the individual needs to assistance? Consider each action and readjust the work assignments. Don't be afraid to put two people on a single task so they can work together and learn from each other.
- List of numbered actions for the next meeting marked against individual names
- Updates to the Kanban Board (screenshot or PDF) for each meeting

Item Tracking

You are to use Planner to maintain a Kanban Board to keep track of all outstanding items, assign names to tasks and to help visualize workload. This must be updated at the end of each group meeting.

Version Control

You are to use version control both individually and as a group.

- It is required that each group member has their own branch.
- The main branch should be your best release candidate
- You should TAG the main branch with each interim release
- You should use git merge to consolidate individual code in the main branch.

See sessions on version control for more details.

Version control is a tool designed to help maintain control and security of your source code, whether this is a programming language or a hardware description language.

The workflow you are required to follow is summarised below. This is intended to help you learn the fundamentals of version control when working as an individual.

3.2 Personal Workflow

The version control for both group and individuals will be scrutinised. Each individual should use the following workflow:

1. Clone the repository if you have not done so already.
    a. If you have made changes on another computer, then fetch and merge the changes
    b. If you have not already done so, create a branch. The name of the branch should contain your name.
2. Make some changes to your code. At strategic points, commit those changes along with the required comment. The strategic points may include:

- When you complete a task or start a new one
- Before you make a potentially breaking change
- When you have finished a work sitting

With each commit, you will be required to add a meaningful comment. If you commit too many changes in one go, it is difficult to capture this in the comments.

3. At the end of each work session, you should push your changes back to GitHub where possible.

4. As you reach each milestone, tag your branch with a meaningful name.

The tutor will be looking at GitHub history for each individual. This is to provide evidence that you have followed this workflow and made a significant contribution.

3.3 Group Workflow

Each group has a shared repository. You should not use any other repository. The main branch should be reserved the latest version of your work (that is ultimately submitted).

- Try to keep the main branch in a working state – it should be your latest and best version. The final commit should be the work you submit
    - If you wish to submit an earlier version, please tag it with the label "FINALSUBMISSION"
- Individual changes will be merged into the main branch. This is potentially breaking, should be done as a group.
- It is important you learn to merge a branch into another, and to resolve any merge conflicts that might occur. If you stick to the guidelines above, you should be able to avoid merge conflicts.

We will be looking for evidence of the above in your Git history.

3.4 Time Management

Your git repository history should be used to demonstrate and evidence good time-management. Sitting down on the day before the deadline and writing everything in one go is likely to result in fatigue and errors. Good practise is to spread the workload over several days or weeks. Your version control commit history can evidence good practice in this respect. Of course, we understand there will be great variation between individuals, and we are looking for:

Work checked in over a reasonable spread of days

Incremental changes to files

Mode of Assessment

All learning outcomes are assessed. Practical project (see above) culminating with a group demo and presentation + Q&A. Upon request, the group will be expected to demonstrate evidence of good project management both at individual and group level, including use of version control, testing, division of labour, quality assurance, documented meetings (online meeting notes in OneNote)

Demonstration and evidence of technical requirements		**75%**
Group management – examined through presentation, meeting notes and version control history		**25%**
Break down of group management mark	Routine and appropriately documented Group Meetings. Evidence of assigned actions, actions from previous meeting, issues raised and a record of people present / apologies for absence	10%
	Evidence of weekly updates to your project plan (Kanban Board) showing work allocation and routine reviews	5%
	Appropriate use of version control.	10%

You must use Microsoft 365, which includes OneNote and Planner

3.5 Peer Assessment

At the end of the work, each group member will be required to score the other group members on their contribution. If there is sufficient evidence that any group member has not contributed to a satisfactory level, then selected group members will be interviewed and assessed individually.

3.6 What to submit

You are to submit the following by the deadline on the DLE:

TABLE 2: TABLE OF WHAT TO SUBMIT

Item	Folder	
Report		You MUST used the report template provided. Do not change the headings or formatting styles.
PCB and Schematic	Hardware	Ensure these can be opened and easily inspected
Final Mbed Studio Files This will be used for your final presentation	Software	Remove the BUILD and mbed library folder first. These are very large and can be recreated easily. Always check that your project files can be downloaded and rebuilt. You will be doing this during the viva.
Final Quartus File and HDL This will be used for your final presentation	Firmware	Always check that your project files can be downloaded and rebuilt. You will be doing this during the viva. Do not include files there are not part of the final build or test, especially if they do not build. You must include all testbenches.
PowerPoint slides	(top level)	These will be the slides used in the viva
group.docx	(top level)	Complete this and include with your submission.

You may include additional documents as evidence as you see appropriate. However, it must be made clear to the examiners how to navigate this. You must explain how you verified each requirement and where the evidence is located. Do not expect examiners to search through pages of documentation looking for this.

3.7 Report

A report template will be provided. You much use this. It is designed to help guide you on how a report should be structured, and what content needs to be contained within. This also provides you with a model to follow in your final year project. Each section has a page limit. Do NOT exceed this.

Frontpage

Table of Contents

- **Analogue Circuit Design & Simulation** (2 Pages)
    - Calculations
    - Simulation (Bandwidth Graph & Voltages at test points)
    - PCB Design (showing component values and test points)
- Signal Sampling (FPGA) (2 pages)
    - HDL code circuit description including test benches (annotated timing diagram and automated tests)
- Real Time Software (2 pages)
    - C code flowchart or pseudocode for main program and what each function does
- Testing (4 pages)
    - Bandwidth measurement (Graph) overlayed on Simulation
    - Data waveform from SPI showing ADC data
    - Testing data to determine peak timing form pulse
    - Show oscilloscope waveform showing peaks and time calculated by your system.
    - Noise and averaging to remove errors
- Results (2 page)
    - Explain differences between simulation and measured performance

    - How accurate is the pulse measurements
    - Output to LCD
- Conclusions (1 page)
    - How well did your system perform
    - Quantitative vs Qualitative (measurements vs good/bad) what would you expect? Is it accurate enough? How could you improve it?
- APPENDIX (No Limit, no marks)
    - Supporting Data
    - Who's done what

3.8 Viva

During the viva you will be required to present evidence of which requirements you have met. This must include (i) your claims (met, partially met, not met), (ii) how you evidenced this (**methodology**) and (iii) a summary of the **results**.

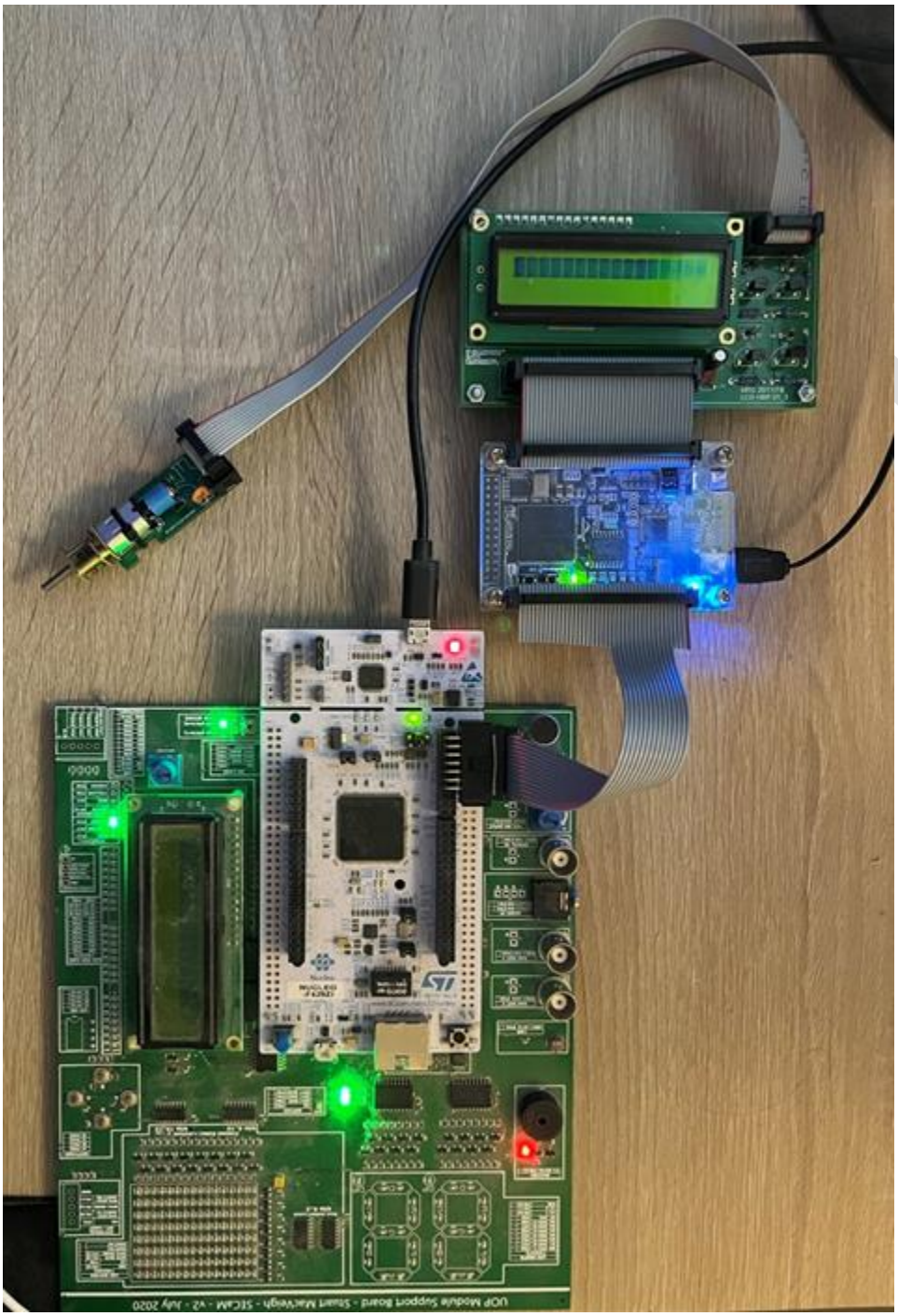
You should also give a demo of any working aspects of your system.

You will be asked questions about both the technical work and how you worked as a group.

You will be asked to share your meeting notes and project planning documents with the tutors.

Appendix A – Connecting the Components

Note the position of the ribbon cable connecting the MCU to the FPGA.



Appendix B – Schematics

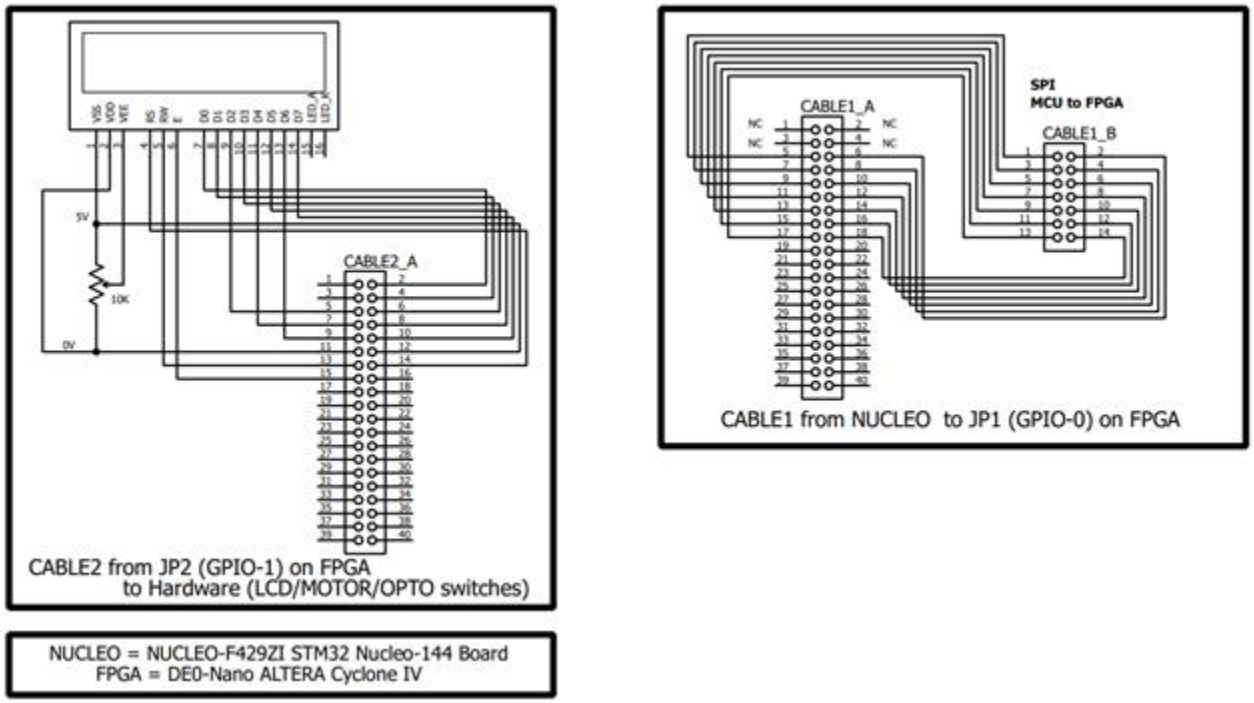


Figure 5. Connection from LCD to FPGA

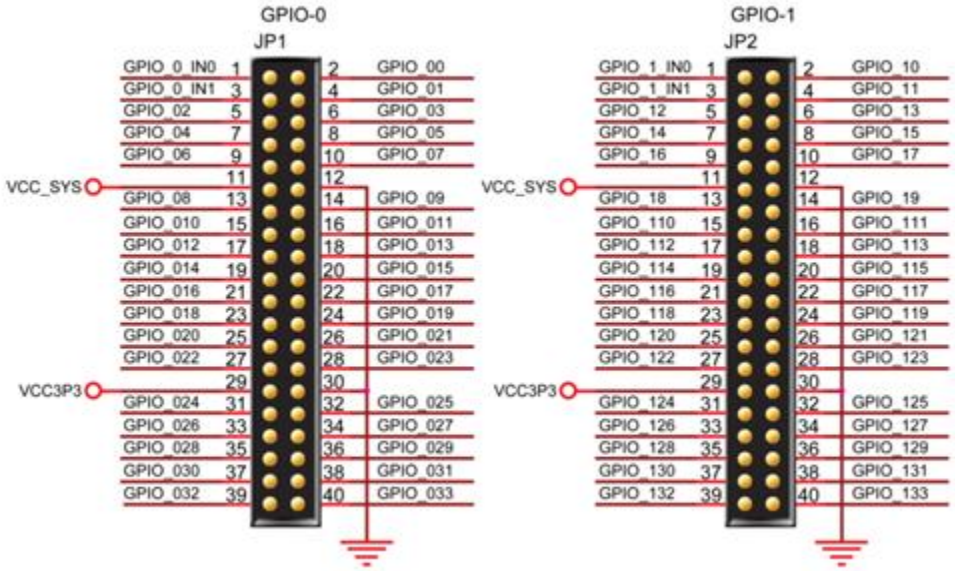


Figure 6. Connections on the DE0-NANO

Appendix B – Pin Assignments

Table 3-6 GPIO-0 Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
GPIO_0_IN0	PIN_A8	GPIO Connection DATA	3.3V
GPIO_00	PIN_D3	GPIO Connection DATA	3.3V
GPIO_0_IN1	PIN_B8	GPIO Connection DATA	3.3V
GPIO_01	PIN_C3	GPIO Connection DATA	3.3V
GPIO_02	PIN_A2	GPIO Connection DATA	3.3V
GPIO_03	PIN_A3	GPIO Connection DATA	3.3V
GPIO_04	PIN_B3	GPIO Connection DATA	3.3V
GPIO_05	PIN_B4	GPIO Connection DATA	3.3V
GPIO_06	PIN_A4	GPIO Connection DATA	3.3V
GPIO_07	PIN_B5	GPIO Connection DATA	3.3V
GPIO_08	PIN_A5	GPIO Connection DATA	3.3V
GPIO_09	PIN_D5	GPIO Connection DATA	3.3V
GPIO_010	PIN_B6	GPIO Connection DATA	3.3V
GPIO_011	PIN_A6	GPIO Connection DATA	3.3V
GPIO_012	PIN_B7	GPIO Connection DATA	3.3V
GPIO_013	PIN_D6	GPIO Connection DATA	3.3V
GPIO_014	PIN_A7	GPIO Connection DATA	3.3V
GPIO_015	PIN_C6	GPIO Connection DATA	3.3V
GPIO_016	PIN_C8	GPIO Connection DATA	3.3V
GPIO_017	PIN_E6	GPIO Connection DATA	3.3V
GPIO_018	PIN_E7	GPIO Connection DATA	3.3V
GPIO_019	PIN_D8	GPIO Connection DATA	3.3V
GPIO_020	PIN_E8	GPIO Connection DATA	3.3V
GPIO_021	PIN_F8	GPIO Connection DATA	3.3V
GPIO_022	PIN_F9	GPIO Connection DATA	3.3V
GPIO_023	PIN_E9	GPIO Connection DATA	3.3V
GPIO_024	PIN_C9	GPIO Connection DATA	3.3V
GPIO_025	PIN_D9	GPIO Connection DATA	3.3V
GPIO_026	PIN_E11	GPIO Connection DATA	3.3V
GPIO_027	PIN_E10	GPIO Connection DATA	3.3V
GPIO_028	PIN_C11	GPIO Connection DATA	3.3V
GPIO_029	PIN_B11	GPIO Connection DATA	3.3V
GPIO_030	PIN_A12	GPIO Connection DATA	3.3V
GPIO_031	PIN_D11	GPIO Connection DATA	3.3V
GPIO_032	PIN_D12	GPIO Connection DATA	3.3V
GPIO_033	PIN_B12	GPIO Connection DATA	3.3V

Table 3-7 GPIO-1 Pin Assignments

Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_1_IN0	PIN_T9	GPIO Connection DATA	3.3V
GPIO_10	PIN_F13	GPIO Connection DATA	3.3V
GPIO_1_IN1	PIN_R9	GPIO Connection DATA	3.3V
GPIO_11	PIN_T15	GPIO Connection DATA	3.3V
GPIO_12	PIN_T14	GPIO Connection DATA	3.3V
GPIO_13	PIN_T13	GPIO Connection DATA	3.3V
GPIO_14	PIN_R13	GPIO Connection DATA	3.3V
GPIO_15	PIN_T12	GPIO Connection DATA	3.3V
GPIO_16	PIN_R12	GPIO Connection DATA	3.3V
GPIO_17	PIN_T11	GPIO Connection DATA	3.3V
GPIO_18	PIN_T10	GPIO Connection DATA	3.3V
GPIO_19	PIN_R11	GPIO Connection DATA	3.3V
GPIO_110	PIN_P11	GPIO Connection DATA	3.3V
GPIO_111	PIN_R10	GPIO Connection DATA	3.3V
GPIO_112	PIN_N12	GPIO Connection DATA	3.3V
GPIO_113	PIN_P9	GPIO Connection DATA	3.3V
GPIO_114	PIN_N9	GPIO Connection DATA	3.3V
GPIO_115	PIN_N11	GPIO Connection DATA	3.3V
GPIO_116	PIN_L16	GPIO Connection DATA	3.3V
GPIO_117	PIN_K16	GPIO Connection DATA	3.3V
GPIO_118	PIN_R16	GPIO Connection DATA	3.3V
GPIO_119	PIN_L15	GPIO Connection DATA	3.3V
GPIO_120	PIN_P15	GPIO Connection DATA	3.3V
GPIO_121	PIN_P16	GPIO Connection DATA	3.3V
GPIO_122	PIN_R14	GPIO Connection DATA	3.3V
GPIO_123	PIN_N16	GPIO Connection DATA	3.3V
GPIO_124	PIN_N15	GPIO Connection DATA	3.3V
GPIO_125	PIN_P14	GPIO Connection DATA	3.3V
GPIO_126	PIN_L14	GPIO Connection DATA	3.3V
GPIO_127	PIN_N14	GPIO Connection DATA	3.3V
GPIO_128	PIN_M10	GPIO Connection DATA	3.3V
GPIO_129	PIN_L13	GPIO Connection DATA	3.3V
GPIO_130	PIN_J16	GPIO Connection DATA	3.3V
GPIO_131	PIN_K15	GPIO Connection DATA	3.3V
GPIO_132	PIN_J13	GPIO Connection DATA	3.3V
GPIO_133	PIN_J14	GPIO Connection DATA	3.3V

Table 3-9 Pin Assignments for ADC

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CS_N	PIN_A10	Chip select	3.3V
ADC_SADDR	PIN_B10	Digital data input	3.3V
ADC_SDAT	PIN_A9	Digital data output	3.3V
ADC_SCLK	PIN_B14	Digital clock input	3.3V

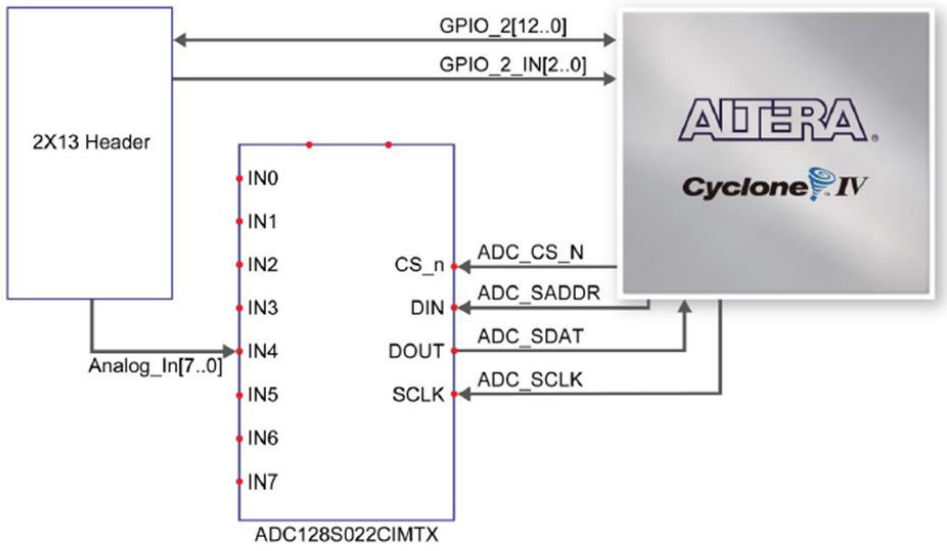


Figure 3-11 Wiring for 2x13 header and A/D converter

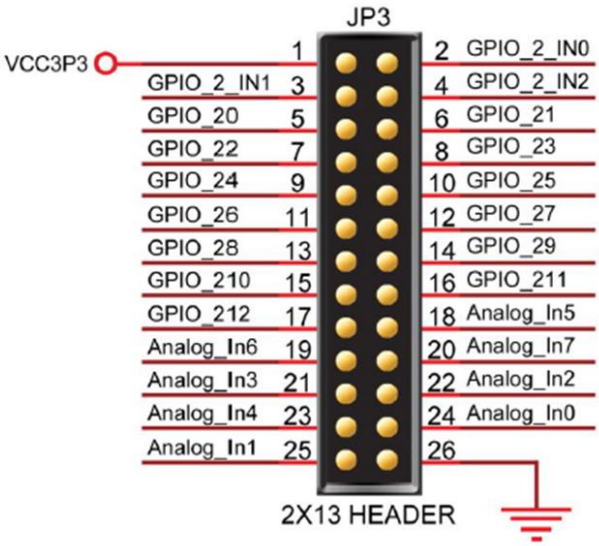
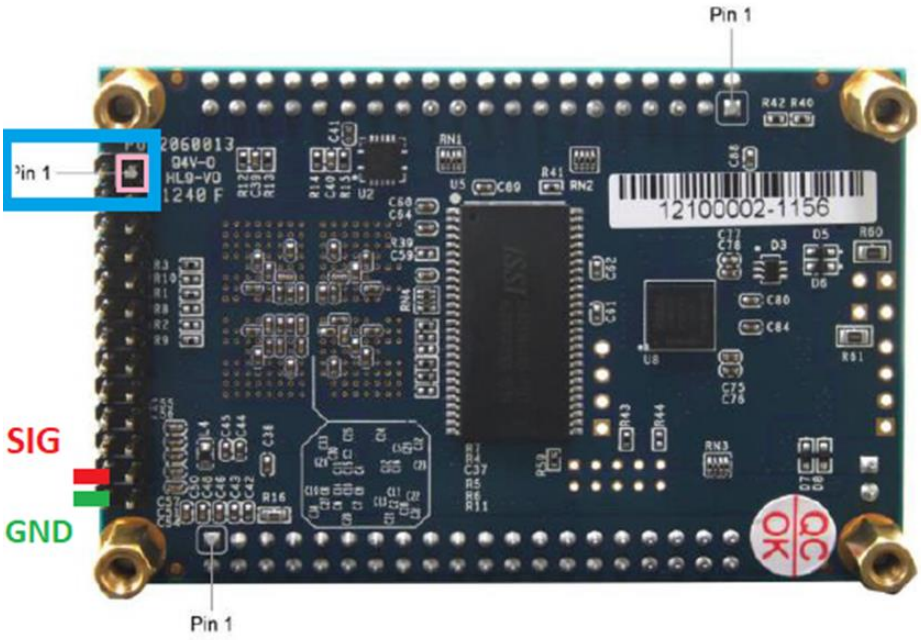


Figure 3-10 Pin distribution of the 2x13 Header



Connections for the Analogue Input shown above JP3

Please Note the location of Pin 1 (Light Blue and Pink) This is the PIN side, i.e. underneath the PCB!

Also Note Signal (RED) and Ground (Green) Pins 24 and 26 respectively

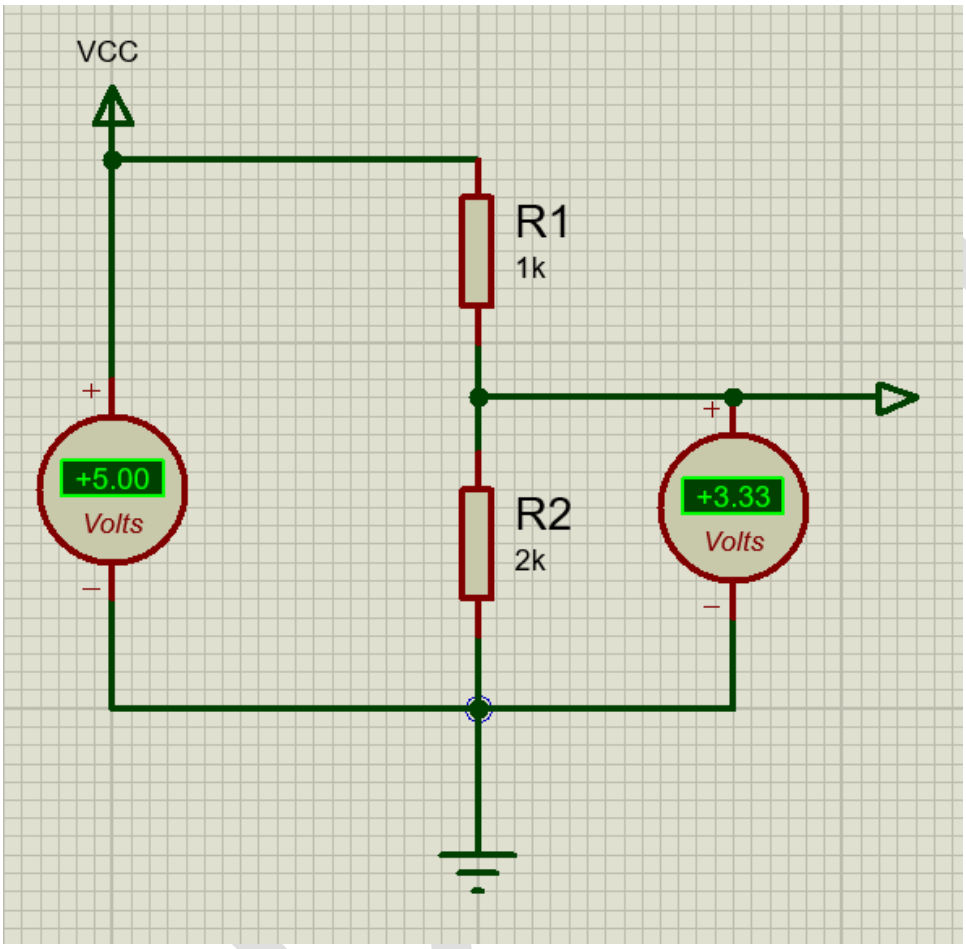
Pin 1 is on the opposite row to Pins 24 and 26. Odd numbers one row and even numbers the other row. See figure 3-10 ‘Pin Distribution’ diagram.

Should be using Analog_In0 signal Input (see Coursework Spec.)

NOTE! The Analogue inputs are NOT 5V Tolerant.

You MUST ensure that the Input Voltage Does not exceed 3V3.

Example below.



NB this circuit isn't ideal but affords some protection.

We know the MCP602 Op-Amp has a 5VDC supply and the output can go within 15 to 20mV of the supply rails. Very close to 0VDC and 5VDC. The output of the MCP602 can supply 20mA from a 5VDC supply as a load effective output impedance of ~250R.

The above circuit is 3K ohms with maximum 5V output would demand 16.7mA So, within the OP-Amp's drive capability. The Input Impedance to the ADC is very high typically <1uA at 3V3 this would be > 3Mohms which is much higher than the potential divider's impedance of 2K//1K//OP-Amp impedance which is approx. (1K+250R)//2K = 1K3. And so should not affect the ADC performance adversely.

Appendix C – LCD

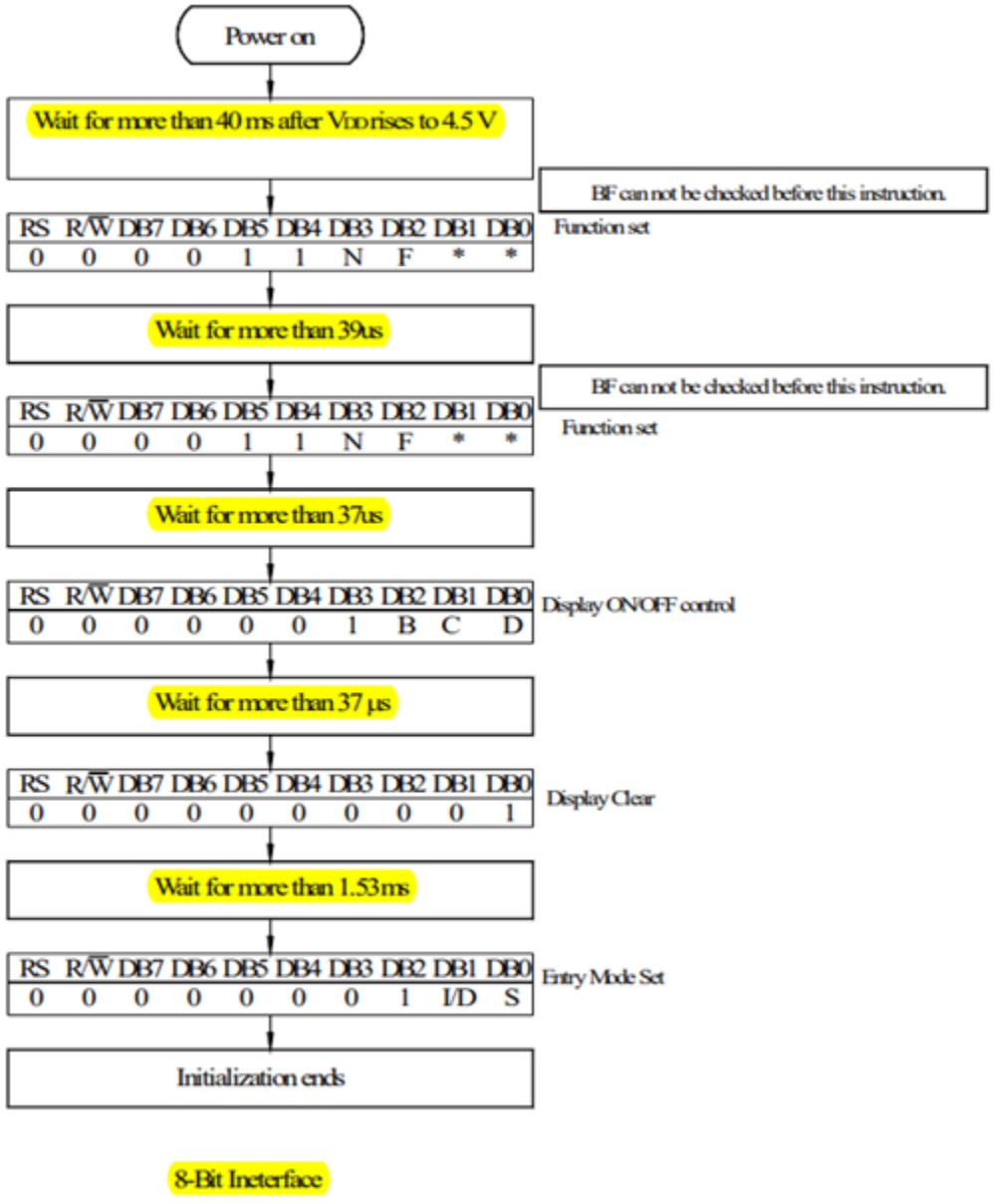


Figure 7. 8-bit Initialization Sequence. N=1 (2-line). F=0 (5x8 display). B=blinking. C=cursor. D=1 (Display). I/D=1 (increment). S=0 (scroll)

Reset Function

Initializing by Internal Reset Circuit

An internal reset circuit automatically initializes the HD44780U when the power is turned on. The following instructions are executed during the initialization. The busy flag (BF) is kept in the busy state until the initialization ends (BF = 1). The busy state lasts for 10 ms after V_{CC} rises to 4.5 V.

1. Display clear
2. Function set:
   DL = 1; 8-bit interface data
   N = 0; 1-line display
   F = 0; 5 × 8 dot character font
3. Display on/off control:
   D = 0; Display off
   C = 0; Cursor off
   B = 0; Blinking off
4. Entry mode set:
   I/D = 1; Increment by 1
   S = 0; No shift

Note: If the electrical characteristics conditions listed under the table Power Supply Conditions Using Internal Reset Circuit are not met, the internal reset circuit will not operate normally and will fail to initialize the HD44780U. For such a case, initial-ization must be performed by the MPU as explained in the section, Initializing by Instruction.

Appendix D – MCU to FPGA (SPI) Connector

Table 3. Suggested connectivity for the MCU-FPGA SPI interface

MCU Signal	MCU Pin	FPGA Pin
CS	PC_6 (D16)	A2 (JP1-Pin 5) GPIO_2
MOSI	PA_7 (D11)	D6 (JP1-Pin18) GPIO_13
MISO	PA_6 (D12)	A6 (JP1-Pin16) GPIO_11
SCLK	PA_5 (D13)	D5 (JP1-Pin14) GPIO_9
GND	CN7, Lower, Pin5	Pin12