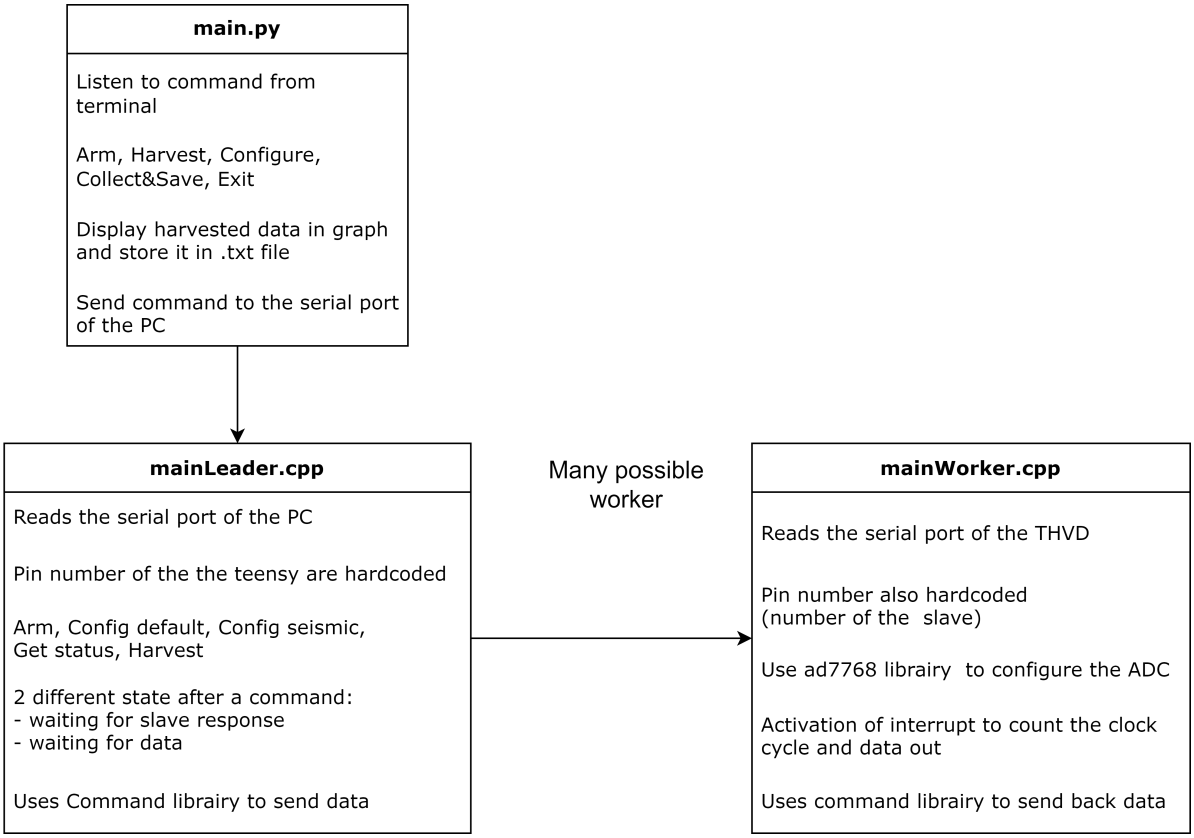
Firmware

In this page you will find details about the Geophysical Open Seismic Hardware firmware, design guidelines and current stage of development. The details about the librairy used in the projects can be found in the command and ad7768 files.

Design guidelines

This basic software is the first prototype of the control software. Only the essentials functions are implemented. This software looks at doing 2 things in particular:

- Ease of use and high-level function to operate the system [main.py and mainLeader].
- High-speed acquisition of seismic signal from the ADC (32 kHz) [mainWorker].



Principal elements in main scripts:

main.py

This script runs on the PC used for the acquisition. It must be launched in the terminal once the leader unit is connect by USB. This script allow the user to enter high level command to perform a basic survey.

- Read the input of the terminal (the serial port and baud rate must be specified, 115200 by default).
- Transform the data from 24 bits to voltage. The values are hardcoded.
- Plot the results of the harvested data (only 3 worker for now).
  The X is the value of the worker you wish to harvest.
- Typical use case:

    - Enter the sampling rate and duration in the terminal.
    - Create a new stack.
    - Use of arm and collect function after each shot (shot count will appear).
    - Use of the show stack function to display individual shots and stacked traces.
    - Create a new stack and carry on.

mainLeader.cpp

This script runs on the leader MCU (Teensy 4.0). The MCU are programmed from the PlatformIO extension in Visual Studio code.

- Read the serial from PC and read the command (Uses the by default serial port to PC and a specific serial port to the RS485 chip).
- If the arm command is detect the trigger interrupt is activated.
- If the trigger is detected, a signal is sent to the workers and the trigerred state is turned off.
- If the harvest data command is detected the flag waitForData is activated.

The script uses the functions of the *command* librairy to communicate with the workers. When the leader waits for a worker response the function *readCommand* of the *command* librairy is used to detect the incoming command type from the worker.

mainWorker.cpp

This script runs on every worker MCU (Teensy 4.0). The MCU are programmed from the PlatformIO extension in Visual Studio code. Each worker must be assigned a unique workerID. This will allow the leader unit to communicate with a single worker at the time. In the future, the workerID attribution would likely be automated.

- Read the serial port of the RS485 communication line.
- Sort the command type with *readCommand* function.
- Configures the adc with the specified configuration.
- If the arm command is detected the interrupt for the clock and for the drdy (dataready from the adc) is activated. The status ready to trig is then activated so the serial port is checked for this particular command.
- Once the ready to trig flag is true, the scripts only waits for a 't' on the serial port. When it sees it, it activated the triggered_state to true and the acquisition time is set to 0. This previous block of code is under a nointerrupt condition.
- When a drdy interrupt is detected the clck count is set to 0.
- the read_ISR function is called every clck count the register the data in a pre defined vector (binary data). The data is read on 3 different pins at 4 MHz. This is why we need a fast MCU (the time required to change pins is long...). Alternatively, one could decide to output every channel of the ADC on the same pins, one after the other. This would mean the MCU as only to read data on a single pin, which is faster.
- At the 33 clck count the data_packet_ready flag is activated and the data is transformed form bit to integer.
- The interrupt are then disable and the triggered status and ready to trig status are set to false.