0.0.1 Outline for this timebox

This timebox includes a project week and a Easter holiday, week 13 & 14.

1. Implement the wishbone interface (Jacob)
2. Design VHDL diagram for Slave 1 (Anders (Jacob))
3. Design VHDL diagram for Slave 2 (Anders (Jacob))
4. Implement VHDL code for slave 1 (Anders (Jacob))
5. Implement VHDL code for slave 2 (Anders (Jacob))
6. Design software for SunTracker (Morten & Henrik)
7. Implement software for SunTracker (Morten & Henrik)
8. Design Voltage Measure PCB for logger (Jacob)
9. Design Current Measure PCB for logger (Jacob)
10. Design Software for Communicator (Henrik)

0.0.2 Development Plan

0.0.3 Results

0.0.3.1 Design & implement code for SunTracker (Morten & Henrik)

Requirements to fulfil
FR 1.a FR 1.c

Theory
The SunTracker function will be operated by the system calling it with 3 variables that can be obtained through the RTC, a fail safe has been made so the system will only access the rest of the code if 4 argument counts has been given.

```
int main(int argc, char *argv[]) {
int hr;
int min;
int day;
if (argc != 4) {
printf("Usage: [hr] [min] [date int]");
exit(1);

}
```

meaning that if the system hasn't send the 4 arguments then the code will write back to the user that he has to send the arguments in order hour - minutes - date int in order to get the sun position corresponding to the time sent.

Implementation
A Makefile is made in order to convert the file type to a hex file that can run in the user space on uClinux on the LPC2478 board.

```
EXEC = SolarPosition
   OBJS = SolarPosition.o sunpos.o

   all: $(EXEC)

   $(EXEC): $(OBJS)
arm-elf-gcc $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS) -elf2flt -lm

SolarPosition.o : SolarPosition.c sunpos.h
arm-elf-gcc -c SolarPosition.c

sunpos.o : sunpos.c sunpos.h
arm-elf-gcc -c sunpos.c

   romfs:
$(ROMFSINST)    /bin/$(EXEC)

   clean:
rm -f $(EXEC) *.elf *.gdb *.o
```

The executeable file that will be made is called "SolarPosition" it is composed of two object files called SolarPosition.o and Sunpos.o which in turn are composed by a header file called sunpos.h and a c file called the same as the object file itself. a command to clean the files made by the Makefile is also made, it will clean the executeable file itself, and all elf, gdb and object files. after the executeable file it made, then it is included in the drivers of uClinux, a image file is then made and transfered to a tffp server, which the LPC2478 board connects to when booting up in order to download the code.

Verification of requirements

0.0.3.2 Relayserver

Requirements to fulfil
NF 3.g NF 3.h NF 3.i BR 4.a BR 4.b BR 4.c

Theory
The Relayserver consists currently of two c files, echoserver.c and client.c.

Echoserver The echoserver first checks if the system gave it the arguments that it needs, which in this case is just a port number that the client will have to match in order to connect to the echoserver. The echoserver will then go through the neccessary steps that is needed to make a socket that can be connected to:

1. Creates a listening socket
2. Set all bytes in the socket address structure to zero and fill in the relevant data members
3. Binds the socket address to the listening socket and calls listen()
4. Enters an infinite loop to respond to client requests and echo input
5. Wait for a connection, then accept() it
6. Retrieve an input line from the conneceted socket then simply write it back to the same socket and write it out on the console on the device
7. Close the connected socket

Client

```
int main(int argc, char *argv[])
{
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
```

The Client.c file requires 3 arguments in order to work, and the counter is below these 3 arguments, then the Client file will respond with the proper syntax to make it work which is "Hostname Port", where the hostname is the ip of the desired device that the client want to connect to, and port is a port decided by both the echoserver and the client.

Implementation

Verification of requirements

0.0.3.3 Design VHDL for Slave 1 & 2 (Anders)

Requirements to fulfil

Theory

Implementation

verification of requirements

0.0.3.4 Design & implement code for Communicator (Henrik)

Requirements to fulfil

Theory

Implementation

verification of requirements

0.0.3.5 Implementation of the wishbone interface (Jacob)

Requirements to fulfil
FR.3.b.i

Theory

0.0.3.6 Design circuit for Log Module (Jacob)

Requirements to fulfil
FR.1.d
NR.1.a

Theory

Decoupling capacitors: Yes decoupling is used to minise the effect of of fast-edges and as you say you dont have any, but they are really needed for analogue IF the supply rails to the chip are noisy, if the rails are noisy then the output signal is going to be more noisy then you would otherwise expect. The other point of putting the de-coupling down is to "de-couple" the track inductance

if it is a big board then there will be significant track inductance and if you try to take a blat of current from that rail, it will locally sag (equally you stop taking higher current it will rise) the cap will attempt to cancel out the indcutive nature of the tracks This track inductance is only a concern when there are sufficiently high frequency currents traveling in it or large steps (edges) of current draw. Digital switching circuits are the best example of this. Low BW opamps, aren't performing any abrupt changes on the power supply.

Instrumentational amplifiers
AMP04
The AMP04 is a single-supply instrumentation amplifier designed to work over a +5 volt to

±15 volt supply range. It offers an excellent combination of accuracy, low power consumption, wide input voltage range, and excellent gain performance.

The equation for the voltage divider is (left side of dotted line):

$$V_o = \frac{R_2}{R_1 + R_2} * V_i \tag{1}$$

OP-amp is configured as a buffer (right side of dotted line), which has a gain of 1, very high input impedance, thus it draws minimum current form the load.

Implementation

verification of requirements