

Computer System Engineering 1

Weather station Documentation

DT4012

Eric Heindl

ID: 19980128-T172

[eric.heindl@gmx.at](mailto:eheindl-its.b2017@fh-salzburg.ac.at)

Esteban Rueda

ID: -19950317-T499

esrueda@ucm.es

WS2019/20

Table of contents

[1. Requirements 3](#_heading=h.3dy6vkm)

[1.1. Recording of temperature 3](#_heading=h.2s8eyo1)

[1.2. Presentation of logged data 3](#_heading=h.17dp8vu)

[1.3. Detection of sun position 3](#_heading=h.26in1rg)

[1.4. Alarm at under or over temperature 3](#_heading=h.lnxbz9)

[1.5. Test- / Fast-Mode 3](#_heading=h.1ksv4uv)

[1.6. Diagram 3](#_heading=h.2jxsxqh)

[1.7. Documentation 3](#_heading=h.z337ya)

[2. Part list 4](#_heading=h.3j2qqm3)

[3. Diagram 4](#_heading=h.1y810tw)

[4. Wiring 5](#_heading=h.2jejfn26blnn)

[4.1. Keypad 5](#_heading=h.2xcytpi)

[4.2. Display 5](#_heading=h.3whwml4)

[4.3. Temperature sensor 6](#_heading=h.qsh70q)

[4.4. Photoresistor and servo motor 6](#_heading=h.1pxezwc)

[5. Code 7](#_heading=h.2p2csry)

[5.1. main.c 7](#_heading=h.147n2zr)

[5.1.1. while loop 8](#_heading=h.3o7alnk)

[5.1.2. temp\_measuring() 8](#_heading=h.23ckvvd)

[5.1.3. LoggedData() 9](#_heading=h.ihv636)

[5.1.4. homescreen() 10](#_heading=h.1hmsyys)

[5.2. pin\_init.c 10](#_heading=h.2grqrue)

[5.3. menu.c 10](#_heading=h.vx1227)

[5.4. tempsensor.c 12](#_heading=h.4f1mdlm)

[5.5. logging.c 12](#_heading=h.19c6y18)

[5.6. lightsensor.c 13](#_heading=h.3tbugp1)

[5.7. display.c 14](#_heading=h.28h4qwu)

[5.8. servo.c 14](#_heading=h.nmf14n)

[6. Possible improvements 14](#_heading=h.37m2jsg)

[7. Sources 15](#_heading=h.46r0co2)

[8. Table of figures 15](#_heading=h.2lwamvv)

# Requirements

## Recording of temperature

Logging of temperature on a fixed time base. Each minute is recorded and stored over a time period of 7 days. When the memory buffer is filled, the recording starts over from the beginning.

## Presentation of logged data

Presentation of recorded data on the LCD by text. Each day is presented by minimum, average and maximum values for temperature.

## Detection of sun position

Find the orientation of the sun with the photo sensor. When requested by the user, enter a mode where the weather station finds the orientation of the sun (can be simulated by any bright light source). The orientation is found by scanning the range by the RC servo and recording the photo sensor output followed by finding the peak of the light and reporting the angle of the servo.

## Alarm at under or over temperature

Alarm. Create an alarm for lower and upper temperature. Limits are adjustable in runtime.

## Test- / Fast-Mode

Test or “fast” mode. Create a test or “fast” mode where time is changed so that one minute in simulated by a second.

## Diagram

Diagram. Create a diagram which illustrated graphically how the system is composed

## Documentation

Documentation of the project. A report which specifies the modules of the system, with a clear connection to this specification, so it is possible to track which specification is solved with which part of the system.

# Part list

The electronic parts we used to implement the weatherstation:

* Microprocessor: Atmel SAM3X8E
* Keypad: AK-804 with 12 buttons
* Photosensor: Photoresistor PGM5xxx-MP
* Temperature sensor: MAX6575L
* Display: NHD-240128WG-ATMI-VZ
* Servo motor: HS-55
* Octal Bus Transceiver: 74HC245

# Diagram

Ein Bild, das Elektronik enthält.

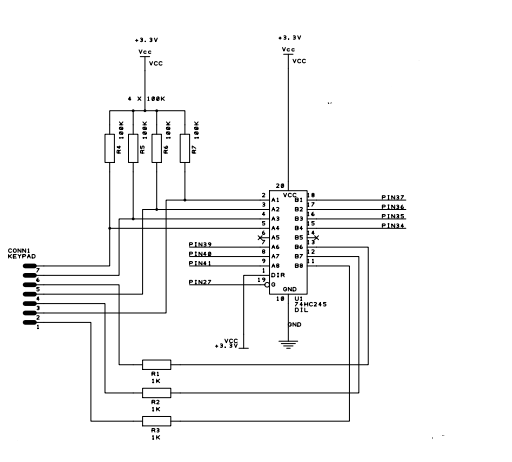
Automatisch generierte Beschreibung

*Figure 1: block diagram*

The block-diagram illustrates the hardware components and how they are connected to each other.

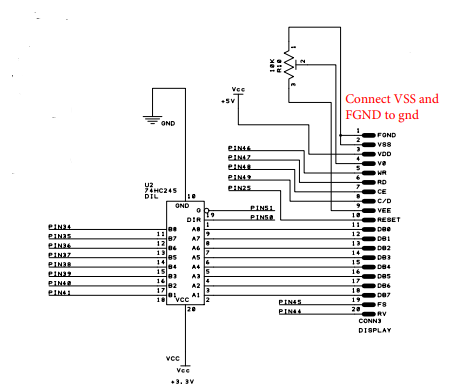
# Wiring

## Keypad



*Figure 2: wiring diagram Keypad*

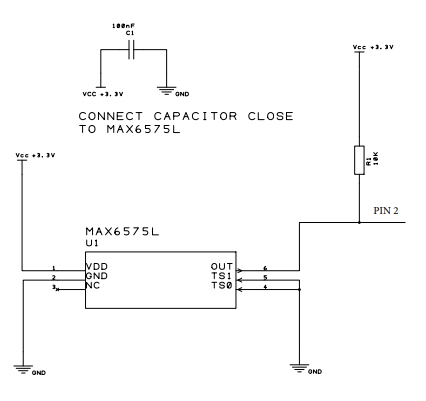
## Display



*Figure 3: wiring diagram Display*

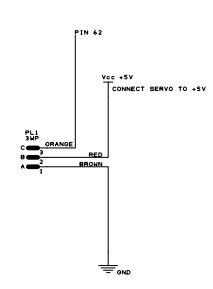
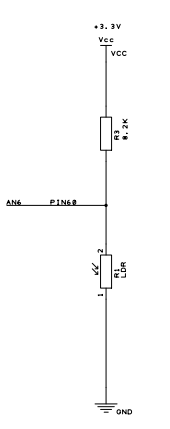
The Keypad and the Display uses the same bus transceiver, therefore they cannot be used at the same time.

## Temperature sensor



*Figure 4: wiring diagram Temperature sensor*

## Photoresistor and servo motor



*Figure 5:wiring diagram Photoresistor and servo motor*

# Code

The code was built up like the hardware laboratories. the code is split into different .c-files with the corresponding header file.

* main.c
* menu.c – menu.h
* Keypad.c – Keypad.h
* lightsensor.c – lightsensor.h
* logging.c – logging.h
* display.c – display.h
* pin\_init.c – pin\_init.h
* servo-c – servo.h
* tempsensor.c – tempsensor.h

## main.c

The main-file is the point where the program starts.

It calls at the beginning the function void main(void):

SysTick\_Config(SystemCoreClock / 1000);

//initialize all needed pins

pin\_init(); // defined in pin\_init.c

//init tempsensor

init\_temp();

measure\_begin();

//init servo

servo\_init();

//init photosensor

initLightsensor();

//initialize display - clear (value from before is stored on display)

init\_display();

//init week array with 0 for every value for every day

DAY\* newday = (DAY\*)malloc(sizeof(DAY));

newday->avg = 0.0;

newday->day = 0;

newday->max = 0.0;

newday->min = 0.0;

int i;

for(i = 0; i < 7; i++){

week[i] = newday;

}

First of all with the function SystTick\_Config() the SysTick is configured to generate an interrupt every millisecond [1]. It calls the function SysTick\_Handeler every time an interrupt appears and increases the variable msTicks.

void SysTick\_Handler(void){

msTicks++;

}

After all the initial function calls, we change the text on the display to “Press # to start!” and go into the main-while loop:

changeLine(1);

char start[] = "Press # to start!";

printString2display(start, sizeof(start));

lastState=0; // for deleting in startMenu() // deleting line 1

## while loop

We always get into this while loop at the beginning or after finishing an action (like detecting the sun). In there we call the function temp\_measunring() (to measure at a fixed time base) and wait for the starting key-button #.

temp\_measuring();

Key\_Btn = read\_keypad(); // reading Keypad // waiting for key 12

if(Key\_Btn == 12){ //# detected - start menu

If the start button is pressed, we open the menu with the function call startMenu() is a menu point chosen we get this value back in the variable action and call this action. Only the sun detection and the show logging data is handled in the main function.

if(action == 6)

{

clearMenu(); // clears only the menu lines

degree = find\_brightestDegree();

light\_intensity = measureBrightness();

homescreen(templogging[pos-1], light\_intensity, degree);

enableTempUpdate=1; // enable fun-call updateTemp in temp\_measuring

lastState=1;

}

else if(action == 2)

{

enableTempUpdate=0;

clearMenu();

loggedData(); // shows the logged data --> exit via Key 10

clearLogging(); // clearing the logging display

homescreen(templogging[pos-1], light\_intensity, degree);

enableTempUpdate=1; // enable fun-call updateTemp in temp\_measuring

lastState=1;

}

## temp\_measuring()

This function will be called in every other function as well, so that we can measure the temperature on a fixed time base. The time base is stored in the variable time. For testing, it can set to a lower value via the function changeFastMode() (Nr. 3 in the menu).

if(tempcount >= time){ //start measurement every time base

tempcount = 0; // increases at SysTick\_Handler()

measure\_begin(); //tempsensor.c

}

After starting the measure, the function waits for the temperature value via an interrupt (TC0\_Handler()). If the value occurred, the value temp\_ready gets true (logical 1).

if(temp\_ready == 1){ //when interrupt detected - calculate temperature

templogging[pos] = calc\_temp(); // save Temperature in templogging

pos++; // increase pos of Templogging

alarmupdate(); // see if Alarm has occured

if(enableTempUpdate==1){ // if homescreen is shown on display

updateTemp(templogging[pos-1]); //updates temperature digits

updateLight(measureBrightness()); // same for Brightness

}

if(pos >= 1440){//if Temperature array full -> fill from beginning -> store day in day array -> calc values for day

pos = 0;

week[day%7] = calculateday(&templogging[0], 20, day);

// calc average, min, max for this Day

day++;

}

temp\_ready = 0; // waiting for new interrupt

}

It’s very important to call this function every loop we pass, because the alarm is also checked in this function (alarmupdate()).

## LoggedData()

This function will be called from the menu and prints last 7 days of the logged data.



*Figure 6: logged data*

week[day%7] = calculateday(&templogging[0], pos, day);

// calc average, min, max for no finished Day

while(i != (day%7))

{

if(((day < 7) && (i < day)) || (day > 7)){ // prints max 6 rows

snprintf(day\_print , 30, " %3i | %3.2f | %3.2f | %3.2f",

(week[i]->day)+1, week[i]->avg, week[i]->min, week[i]->max);

changeLine(j);

printString2display(day\_print, sizeof(day\_print));

j++;

printed\_days=j-3; //for deleting // start from 3 to J

}

i++;//increment i %7

i = i % 7; //only 7 elements in array

}

//at least 1 row to print

snprintf(day\_print , 30, " %3i | %3.2f | %3.2f | %3.2f",

(week[i]->day)+1, week[i]->avg, week[i]->min, week[i]->max);

changeLine(j);

printString2display(day\_print, sizeof(day\_print));

int key =0;

do{

key = read\_keypad();

temp\_measuring(); //to ensure measurement every second

}while(key != 10); // press 10 to exit to homescreen

## homescreen()

The homescreen is the main screen and is called by the sun detect action or if you close the logged data screen.



*Figure 7: homescreen*

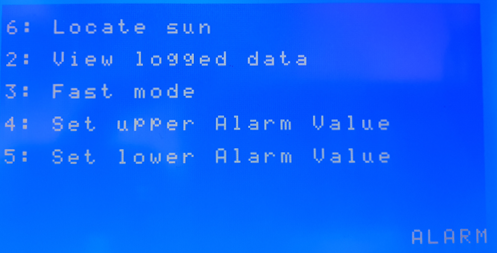
## pin\_init.c

The pin\_init() is the first function which is called in the main. It is used to enable the clock and the pins needed by other functions, like the control pins for keypad and display, pins for the data bus and the sensors.

The file also includes a function to set the Data bus as OUTPUT or INPUT (set\_data\_bus\_direction()).

## menu.c

In the menu all the actions can be chosen. It prints all the possible options and wait for a button input.



*Figure 8: menu*

int startMenu(void){

enableTempUpdate=0;

...

int key = waitForKey();

switch(key){

case 6 : return 6; break; //sun detection

case 2 : return 2; break; //logged data

case 3 : changeFastMode(); break;

case 4 : alarmHigher(); break;

case 5 : alarmLower(); break;

}

return 0;

}

int waitForKey(void){

int key = read\_keypad();

while((key < 2) || (key > 6)){

key = read\_keypad();

temp\_measuring();

}

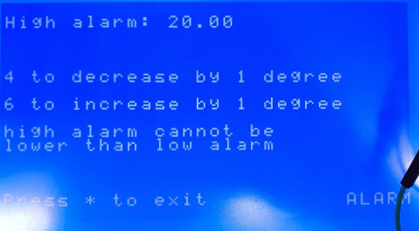
return key;

}

The file also implements the functions to change the alarm values.

void alarmHigher(void)

void alarmLower(void)

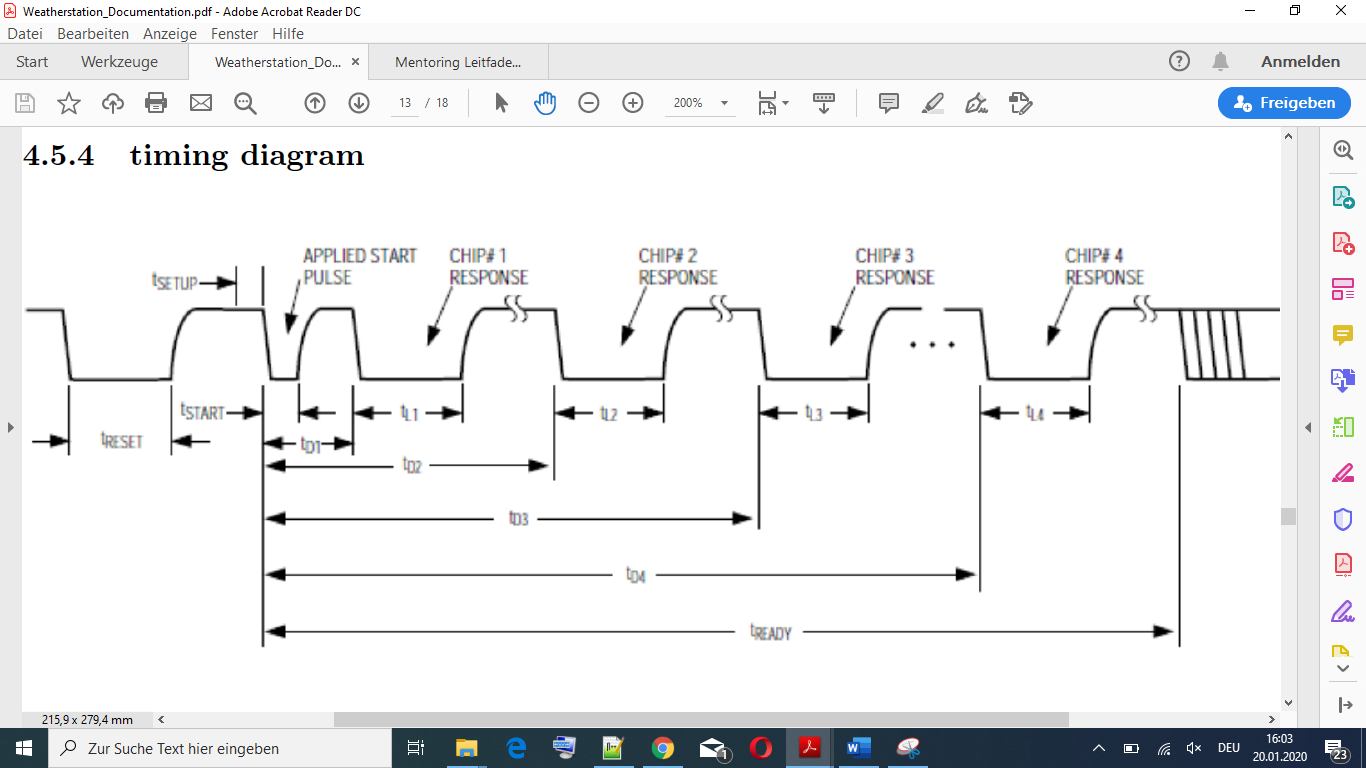


*Figure 9: set high alarm value*

## tempsensor.c

This function measures and calculates the temperature from the MAX6575L.

The measure starts with the function measure\_begin(). This code reset the counter make the Pin as OUTPUT and make a start pulse. Afterwards the Pin is set as INPUT again.



*Figure 10: Timing diagram temperature sensor*

If an interrupt is detected, (with the TCOHandler()) the temperature is calculated:

float calc\_temp(void)

{

\*AT91C\_TC0\_IDR = AT91C\_TC\_LDRBS; //disable interrupts

//calc temperature and return

int rb = \*AT91C\_TC0\_RB; // stores time of interrupt

int ra = \*AT91C\_TC0\_RA; // stores time of start pulse

double ret = (rb - ra);

// T in us (tried a 1ms pulse and read the counter)

ret = ret / 1.895;

ret = ret / (5.0);

\*AT91C\_TC0\_RB = 0x0;

\*AT91C\_TC0\_RA = 0x0;

return ret/(22.4)-273.15; // calculated val 22.4 by measuring

}

## logging.c

In the logging array all the temperature data is stored. The file logging.c has all the calculations:

double calcMin()

double calcMax()

double calcAvg()

DAY\* calculateday(int\* arr, int size, int d){

DAY\* newday = (DAY\*)malloc(sizeof(DAY));

newday->min = calcMin(arr, size);

newday->max = calcMax(arr, size);

newday->avg = calcAvg(arr, size);

newday->day = d;

return newday;

}

The function calculateday() is always called if a whole day is recorded (logging array full).

It creates a new Pointer to a DAY (DAY\*) and calculates the minimum, maximum and average from the logging array.

We chose first a DAY-Pointer to make it easier to iterate through the array. Therefore you can just get the address of the first element and increase it by 1 to get the next one.

It would be possible as well to store directly the DAYs into the array without a pointer.

typedef struct day\_values{

double min;

double max;

double avg;

int day;

} DAY;

## lightsensor.c

This file measures the brightness and finds the brightest degree of the servo motor.

The function find\_brightestDegree() measure the brightness every degree (0° - 180°) and returns the brightest one. That’s how you find the sun location.

int find\_brightestDegree(void){

double brightest = measureBrightness();

double brightness\_cur;

for(i = 0; i<180; i++){

if((i%20) == 0) //to ensure measurement every second

temp\_measuring();

move\_to\_position(i);

light\_intensity= measureBrightness();

if((brightness\_cur=measureBrightness())<brightest){

brightest=brightness\_cur;

light\_intensity= brightness\_cur;

degree=i;

}

msdelay(2000);

}

move\_to\_position(degree);

return degree;

}

## display.c

In this file the display is written with the functions printString2display() and print\_data\_2\_display()

To increase the clearing, we create for every screen its own clearing function:

void clearDisplayLine(int pixelAmount,int line);

void clearMenu(void);

void clearLogging(void);

void clearAlarmMenu(void);

## servo.c

This file contains the function move\_to\_position().

By trying, we realized the servo uses a quadratic function to get from the value to the degrees.

With 3 Values we could create this function:

int move\_to\_position(int pos)

{

float val = 0;

float abs\_pos = (float)pos;

//explanation:

//0 degrees = (val=900) --> offset

//90 degrees = (val=3900) --> middle pos

//180 degrees = (val6300) -->max

val=(-0.037)\*abs\_pos\*abs\_pos+36.6\*abs\_pos+900;

\*AT91C\_PWMC\_CH1\_CDTYR = (int)val; //set value of pwm (CDTY = CDTYR)

return 0;

}

# Possible improvements

* Freeing of Data: to improve the memory storage, the overwritten days should be freed

# Sources

[1]: <https://www.keil.com/pack/doc/cmsis/Core/html/group__SysTick__gr.html>, 19.01.2020

[2]: <https://bb.hh.se>, 19.01.2020

# Table of figures

[Figure 1: block diagram 4](#_heading=h.4i7ojhp)

[Figure 2: wiring diagram Keypad 5](#_heading=h.1ci93xb)

[Figure 3: wiring diagram Display 5](#_heading=h.2bn6wsx)

[Figure 4: wiring diagram Temperature sensor 6](#_heading=h.3as4poj)

[Figure 5:wiring diagram Photoresistor and servo motor 6](#_heading=h.49x2ik5)

[Figure 6: logged data 9](#_heading=h.32hioqz)

[Figure 7: homescreen 10](#_heading=h.41mghml)

[Figure 8: menu 11](#_heading=h.3fwokq0)

[Figure 9: set high alarm value 12](#_heading=h.1v1yuxt)

[Figure 10: Timing diagram temperature sensor 12](#_heading=h.2u6wntf)