2. MAI 2017

DEVELOPMENT OF A LOW-COST SENSOR SETUP FOR SPATIAL SNOW DEPTH MEASUREMENTS

ASSOZ. PROF. DI DR. STEFAN ACHLEITNER
FLORIAN BIRKNER, BSC

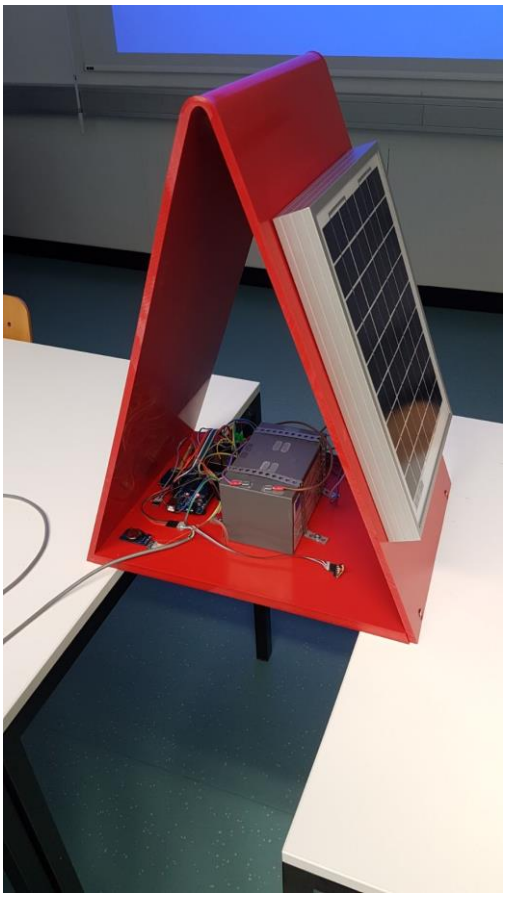
Unit of Hydraulic Engineering - Institute of Infrastructure Faculty of Engineering Science – University of Innsbruck

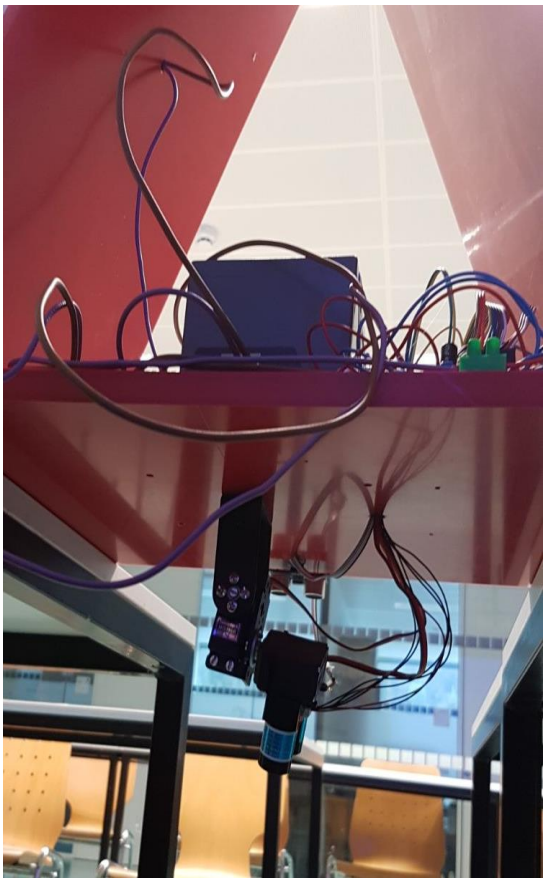
ING. BIRKNER GERT, BED
FUETSCH ARTHUR
BARIC ALEN
MATHIAS WICHENTH.-STERNBACH
NIKOLAS PACIK
MARKUS PLANKENSTEINER
CHRISTOPH MELLAUNER
ALEXANDER VUKOVIC
STEFAN SCHWARZENBERGER

4bINFT
Tiroler Fachberufsschule für Elektrotechnik, Kommunikation und Elektronik - Innsbruck

Table of Contents

List of Figures





Introduction

The development of a Low-Cost Sensor Setup for the Spatial Measurement of Snow Depth was a cooperative project between the Unit of hydraulic Engineering, University of Innsbruck and the Tiroler Fachberufsschule für Elektrotechnik, Kommunikation und Elektronik (TFBS EKE). Eight students of TFBS EKE developed an autonomous system to collect data of various sensors with preferably cheap standard components and make them available via android app. The time for this course, which lasted 9 weeks was very limited and some of the tasks planed could not be completed.

It was a great challenge to carry out a project like this. We started from scratch by determining all the necessary features and working up the user stories in an agile project management environment. In weekly sprint sessions, we defined the tasks to be dealt with and the students responsible for each.
This manual describes characteristics and usage of the modules by taking up each one individually. As main controller, an Atmel ATmega2560 on an Arduino board was used. Instead of Arduino libraries we coded in plain C++ on Atmel Studio. The supply Voltage of each module is switched off to save power in standby mode. Before every use of the laser-servo motors the battery status gets checked to prevent complete discharge during the operation. The necessary settings can be entered via android app and are then transmitted to the µControllers EEPROM. In case of a major malfunction, a watchdog timer initiates a system reset and the µController restarts.

Even though the students had 45 hours of school per week they put in extra time and effort to get this far. It was an extraordinary experience for everyone to be part of a project this complex. That it will actually be used by professionals was a great motivation along the way.

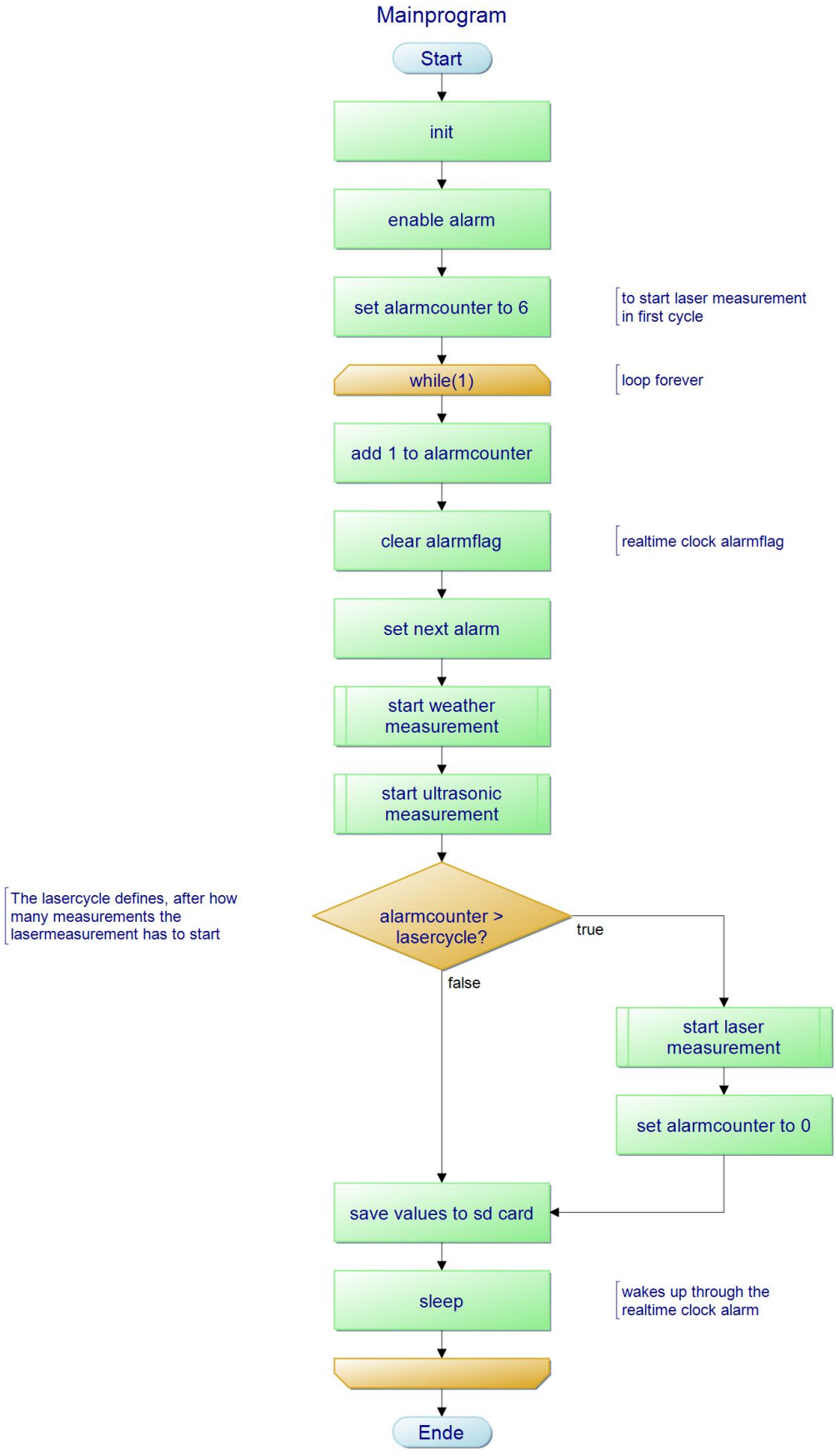
Main Sequence

The main sequence of the Snow Depth measurement system is responsible for initializing and executing all the instruments.

The main program sequence runs through the following steps:

- Turn watchdog on
- Set the next alarm
- Run the environment measurement
- Run the ultrasonic measurement
- Check if it is time for a laser measurement and execute one if due
- Turn watchdog off
- The controller goes into sleep mode

In case the laser measurement returns an error, the next laser measurement is scheduled around noon.
The main program turns on the watchdog, in case the program gets stuck in a subroutine.
The watchdog is turned off before the controller goes into sleep mode.


Mainprogram
Start
init
enable alarm
set alarmcounter to 6
to start laser measurement in first cycle
while(1)
loop forever
add 1 to alarmcounter
clear alarmflag
realtime clock alarmflag
set next alarm
start weather measurement
start ultrasonic measurement
The lasercycle defines, after how many measurements the lasermeasurement has to start
alarmcounter > lasercycle?
true
false
start laser measurement
set alarmcounter to 0
save values to sd card
sleep
wakes up through the realtime clock alarm
Ende

EEPROM

The EEPROM can store 4 kilobytes of data and is used to store general parameters.

The most important parameters are:

- Height of the device
- The duration between ultrasonic measurements
- The duration between laser measurements
- The duration between the laser movement and the laser measurement
- The matrix for the laser measurement as a two-dimensional array

On first use, the user must insert those parameters.
The other devices can read those parameters from the EEPROM.

Real Time Clock

General Information

The “Real Time Clock” (RTC) is used for waking up the controller during sleep times.
The alarm interval gets set automatically at the start of the program based on the alarm cycle the user defines at installation of the snow depth measurement system inside the app.
Once an alarm interval point is reached, the RTC sends a signal to the alarm pin and wakes the controller via an external interrupt. This interrupt causes the initiation of the main program sequence.

Battery

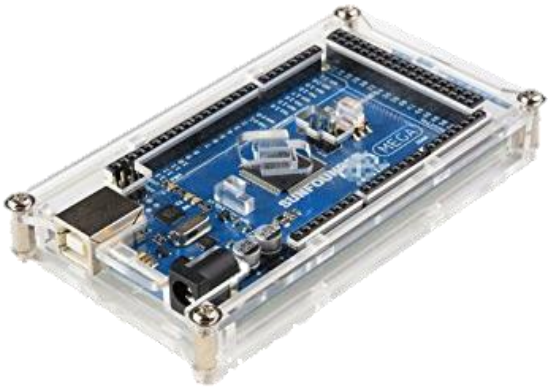
The RTC has its own power source and does not get charged by the solar panel. In normal circumstances the battery should last for at least 10 years.

General Information

For the weather measurement we use the Bosch BME280 temperature sensor. This sensor can measure temperature, air pressure and humidity.
For the connection to the sensor we use the I^2C Interface on our Atmega 2560 Board.
The program writes the measured values into a file, which can be displayed in the app.

Atmega 2560 Board (Atmel Corporation, 2014)

The Atmega 2560 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the Atmega2560 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.



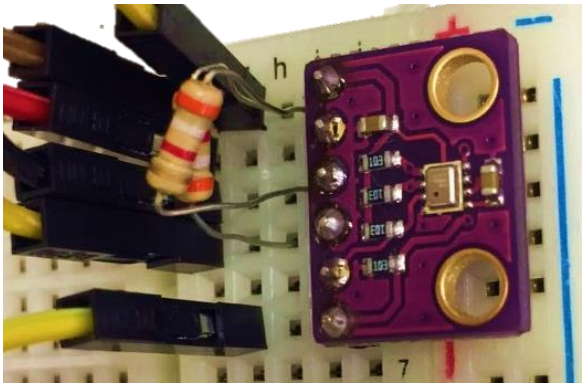
Pic. 1: Atmega 2560 Board

GY-BM E/P 280

To use the GY-BM E/P 280 module the connection pins got soldered to the GY-BM E/P 280 module. The dimensions of the module are 11 mm x 15 mm. The I^2C-address of the Module is 0X77 if the SD0 Port is connected to the VCC Port on the Module (see Pic. 1). The CSB Port is connected to the VCC port (on the Module), which means that the connection works via I^2C. We have connected 2 4,7 kΩ pullup resistors from SCL and SDA to VCC.

Bosch BME280 sensor (Bosch, 2017)

The BME280 is a combined digital humidity, pressure and temperature sensor based on proven sensing principles. The sensor module is housed in an extremely compact metal-lid LGA package with a footprint of only 2.5 × 2.5 mm^2 with a height of 0.93 mm. Its small dimensions and its low power consumption allow the implementation in battery driven devices such as handsets, GPS modules or watches. The BME280 is register and performance compatible to the Bosch Sensortec BMP280 digital pressure sensor.



Pic. 2: GY-BM E/P 280 with Bosch BME280 Sensor

Key features

• Package	2.5 mm x 2.5 mm x 0.93 mm metal lid LGA
• Digital interface	I^2C (up to 3.4 MHz) and SPI (3 and 4 wire, up to 10 MHz)
• Supply voltage	VDD main supply voltage range: 1.71 V to 3.6 V VDDIO interface voltage range: 1.2 V to 3.6 V

- Current consumption — 1.8 µA @ 1 Hz humidity and temperature
  2.8 µA @ 1 Hz pressure and temperature
  3.6 µA @ 1 Hz humidity, pressure and temperature
  0.1 µA in sleep mode
- Operating range — -40…+85 °C, 0…100 % rel. humidity, 300…1100 hPa
- Humidity sensor and pressure sensor can be independently enabled / disabled
- Register and performance compatible to Bosch Sensortec BMP280 digital pressure sensor
- RoHS compliant, halogen-free, MSL1

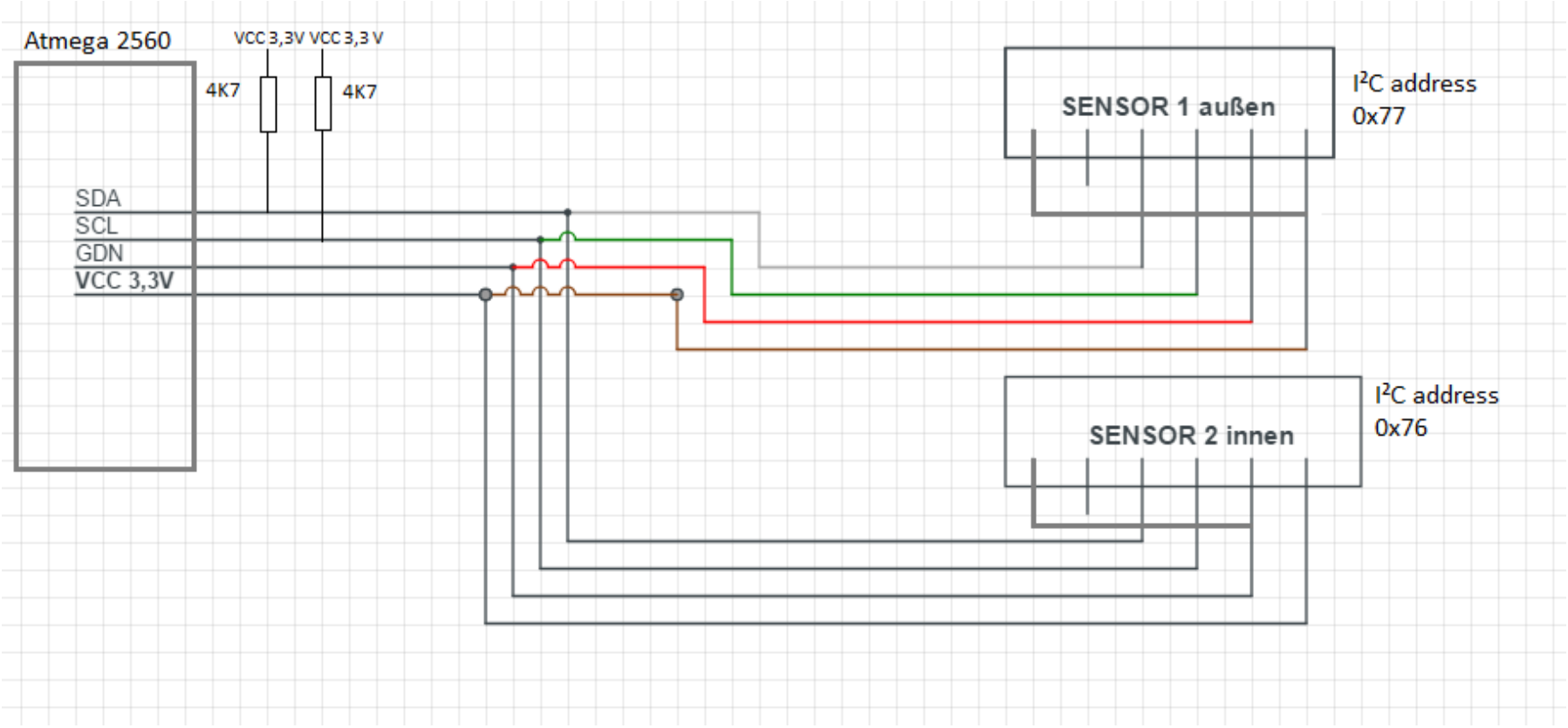
Key parameters for humidity sensor

- Response time — 1 s
- Accuracy tolerance — ±3 % relative humidity
- Hysteresis — ±1% relative humidity

Key parameters for pressure sensor

- RMS Noise — 0.2 Pa, equiv. to 1.7 cm
- Offset temperature coefficient — ±1.5 Pa/K, equiv. to ±12.6 cm at 1 °C temperature change

Temperature Sensor wiring



Pic. 3: Temperature Sensor wiring

Weathersensor Code

The readData-function reads the BME280 register:

0X77 → sensor address

0xF7 – 0xF9 → pressure

0xFA – 0xFC → temperature

0xFD – 0xFE → humidity

After reading the BME 280 register the read-function saves the values in uint32_t arrays.

Functionality of I^2C

To read the I^2C interface, you must observe the following:

1) First start the I^2C connection in write mode
2) write the required register
3) stop the I^2C connection
4) start I^2C in read mode
5) (repeated) readout data from interface
6) stop I^2C connection

When the data is read out, an auto-increment is used, which means that you can read out the data one at a time.

Procedure

To use the BME280 Sensor you have to connect the VCC Port with 3,3V. Connect the SD0 Port to VCC 3,3V to set the Module address to 0x77. For the connection to the I^2C Interface we used the I^2C-master library. To connect to the sensor set the oversampling for humidity, pressure and temperature.

Set the humidity oversampling as in the example below

Set humidity oversampling

1) Start I^2C connection and write crtl_hum register
2) Stop I^2C connection
3) Start I^2C connection and read out the value and write it in a variable
4) Stop the I^2C connection
5) Set the first 3 bits (0-2) to set crtl_hum according to the original datasheet
6) Start the I^2C connection and write the value in the crtl_hum register
7) Stop the I^2C connection

```
//*********************************** Humidity oversampling
i2c_start((address << 1) + I2C_WRITE);
i2c_write(0xF2);
i2c_stop();
i2c_start((address << 1) + I2C_READ);
uint8_t F2var = i2c_read_nack();
i2c_stop();
F2var &= ~((1<<2) | 1<<1);
F2var |= (1<<0);
i2c_start((address << 1) + I2C_WRITE);
i2c_write(0xF2);
i2c_write(F2var);
i2c_stop();
```

Set temperature and pressure oversampling

1) Start the I^2C connection and write the crtl_meas register
2) Write the temperature and pressure oversampling data in the register
3) Stop the I^2C connection

```
i2c_start((address << 1) + I2C_WRITE);
i2c_write(0xF4);
i2c_write(BME280_TempPressMode1);
i2c_stop();
```

Readout raw data and compensation data

Read raw data

```
i2c_start((address << 1) + I2C_WRITE);                  // send I2C register address
i2c_write(BME280_TEMPDATA);                             // send temperature register address
i2c_stop();                                             //Stop I2C
_delay_ms(6);                                           //set delay to convert (6 ms)
i2c_start((address << 1) + I2C_READ);                   // start I2C and read temperature

uint8_t temp[3];                                        //create array where temp can be saved
uint8_t press[3];                                       //create array where press can be saved
uint16_t humid[2];                                      //create array where humid can be saved

press[0] = i2c_read_ack();                              //Write readed data in pressure array
press[1] = i2c_read_ack();
press[2] = i2c_read_ack();
temp[0] = i2c_read_ack();                               //Write readed data in temperature array
temp[1] = i2c_read_ack();
temp[2] = i2c_read_ack();
humid[0] = i2c_read_ack();                              //Write readed data in humidity array
humid[1] = i2c_read_nack();
i2c_stop();                                             //stop I2C

adc_T = read24(0xFA);
adc_P = read24(0xF7);
adc_H = humid[0];

CompensateData();                                       //start function CompensateData()
```

Read compensation data

```
i2c_start((address << 1) + I2C_WRITE);  // start I2c
i2c_write(0xF4);
i2c_write(BME280_TempPressMode1);
i2c_stop();
_delay_ms(6);
i2c_start((address << 1) + I2C_WRITE);
i2c_write(BME280_TEMPCOMP);
i2c_stop();
i2c_start((address << 1) + I2C_READ);
uint16_t dig_t[3];
uint16_t dig_P[9];
uint16_t dig_H[5];

dig_t[0] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_t[1] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_t[2] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[0] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[1] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[2] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[3] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[4] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[5] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[6] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[7] = i2c_read_ack() + (i2c_read_ack() << 8);
dig_P[8] = i2c_read_ack() + (i2c_read_ack() << 8);
i2c_read_ack();
dig_H[0] = i2c_read_nack();
i2c_stop();

i2c_start((address << 1) + I2C_WRITE);                  // start I2c in
i2c_write(0xF2);
i2c_write(BME280_HUMCOMP);
i2c_stop();
i2c_start((address << 1) + I2C_READ);
dig_H[1] = i2c_read_ack() + ((uint16_t)i2c_read_ack() << 8);
dig_H[2] = i2c_read_ack();
dig_H[3] = ((uint16_t)i2c_read_ack() << 4) + (i2c_read_ack() & 0b00001111);
dig_H[4] = (i2c_read_ack() >> 4) + ((uint16_t)i2c_read_ack() <<4);
dig_H[5] = i2c_read_nack();
i2c_stop();
```

Calculate Data

```
//*********************** Temperatur berechnen *********************************
int32_t var1temp, var2temp;
adc_T >>= 4;
var1temp  = ((((adc_T>>3) - ((int32_t)dig_t[0] <<1))) * ((int32_t)dig_t[1])) >> 11;
var2temp  = (((((adc_T>>4) - ((int32_t)dig_t[0])) * ((adc_T>>4) - ((int32_t)dig_t[0]))) >> 12) *
((int32_t)dig_t[2])) >> 14;
t_fine = var1temp + var2temp;
float T  = (t_fine * 5 + 128) >> 8;
temperature = T;

//*************************** Pressure Berechnen *********************************
int64_t var1press, var2press, p;
adc_P >>= 4;
var1press = ((int64_t)t_fine) - 128000;
var2press = var1press * var1press * (int64_t)dig_P[5];
var2press = var2press + ((var1press*(int64_t)dig_P[4])<<17);
var2press = var2press + (((int64_t)dig_P[3])<<35);
var1press = ((var1press * var1press * (int64_t)dig_P[2])>>8) +
((var1press * (int64_t)dig_P[1])<<12);
var1press = (((((int64_t)1)<<47)+var1press))*((int64_t)dig_P[0])>>33;
p = 1048576 - adc_P;
p = (((p<<31) - var2press)*3125) / var1press;
var1press = (((int64_t)dig_P[8]) * (p>>13) * (p>>13)) >> 25;
var2press = (((int64_t)dig_P[7]) * p) >> 19;
p = ((p + var1press + var2press) >> 8) + (((int64_t)dig_P[6])<<4);
pressure = p/256/100;
```

```
//*********************** Berechnung Humidity*******************
double var_H;
adc_H = read8(0XFD);

var_H = (((double)t_fine) - 76800.0);
var_H = (adc_H - (((double)dig_H[3]) * 64.0 + ((double)dig_H[4]) / 16384.0 * var_H)) *
(((double)dig_H[3]) / 65536.0 * (1.0 + ((double)dig_H[5]) / 67108864.0 * var_H *
(1.0 + ((double)dig_H[2]) / 67108864.0 * var_H)));
var_H = var_H * (1.0 - ((double)dig_H[0]) * var_H / 524288.0);
humidity = adc_H;
```

Ultrasonic Sensor

The Ultrasonic sensor used in the Snow depth measurement is a HC-SR04 manufactured by Micropik. It is a low-end measurement device consuming minute amounts of power so as not to drain the battery quickly. Its main use is to measure the snow depth every x (configurable inside the mobile app) Minutes and should work even when power is low and the laser measurement is turned off due to power restrictions. The exact delay between measurements is specified during the installation process of the device.

HC-SR04 reading

Because Ultrasonic wave distribution speed varies depending on the temperature we execute a environmental measurement before activating the Ultrasonic sensor.

To ensure that a new measurement is made a 10µs signal is supplied to the trigger input. 250µs after the falling edge of this signal, the sensor then sends out 8, 40kHz ultrasonic bursts which reflect of solid surfaces. If no echo is detected, the module will return a 200ms signal, whereas it will return the duration from burst to echo. The distance of the object is calculated by measuring the length of the returned signal from the echo pin which is multiplied in a formula with the before measured temperature.

Exactly 10 measurements are made, from which the imprecise get filtered out. The left distances are then averaged and returned to the main function.

HC-SR04 technical data (ElecFreaks, 2017)

- **Minimum range:** 2cm
- **Maximum range:** 4m
- **Minimum temperature:** -20°C (Not tested for -30°C and will cause inaccuracies. A different Ultrasonic device may be used)

SD Card reader

The SD Card reader used in the Snow depth measurement is a generic “Micro SD card mini TF card reader” module interfacing over SPI, which means we control dataflow over the specified MOSI and MISO pins. The module is used to store collected data from measurements.

File structure

To ensure usability we separated the collected Data into 3 files all using the csv file format. Every file contains at least an ID and a Timestamp. Additionally we add extra data depending on the file:

- **ED (Errordata)** - Stores all Errors that were encountered during the usage of other modules and therefore has extra “Module” and “Errormessage” columns
- **LD (Laserdata)** - Stores the distance and LaserID of each measured point in a laser measurement. It should be noted that LaserID and ID are two different fields
- **GD (Genericdata)** - Stores the data collected using every data collecting module (except Laser module) i. e. Ultrasonic distance measurements and temperature, humidity and pressure of the weather sensor. Inside the File are extra “Module” and “Data” fields

SD reading and writing

In order to read and write files we implemented FAT32 for avr using a free, already available library which uses standard SD and FAT commands to create and read files. Additionally, we also implemented wrapper functions to ensure easy reading and writing to and from our file structure. This library also handles creation and detection of the files and adds the respective headerdata.

SD module technical data (Datalogger, 2017)

- Supports Micro SD and Micro SDHC cards
- Easy SD insertion mechanism
- 6 Pins for SPI data transfer which are used as follows
    - **GND:** Ground connection
    - **VCC:** 5V Power supply
    - **MISO:** Used to receive data from the SD card module
    - **MOSI:** Used to send data to the SD card module
    - **SCK:** Clock signal used for synchronisation
    - **CS:** This is the chip select signal pin, which is not used as we only have one module interfacing over SPI

Bluetooth Module

The Bluetooth module used in the Snow depth depth is a zs-040 manufactured by Bolutek. It is a low-end Bluetooth device consuming low amounts of power so as not to drain the battery quickly. Its main use is to deliver data between the mobile app and the snow depth measurement device. In order to exchange the data correctly between the mobile app and the device, a self-made communication protocol is used.

Communication protocol

Our self-made communication protocol starts by sending the data. If everything went fine and the whole data was sent successfully, two end characters ("\" & "x") will be transmitted. The end characters are used to confirm the correct arrival of the data.

Zs-040 pin usage

- **STATE:** Not in use
- **RXD:** Receives data from an external device
- **TXD:** Transmits data to an external device
- **GND:** Used to ground the Bluetooth module
- **VCC:** 5V Power supply
- **EN:** Not in use

Servo motors

The servo motors used in the Snow depth measurement are two MG996R High Torque Metal Gear Dual Ball Bearing Servos which have a range from 30° to 120°. One is responsible for the horizontal and the other for the vertical movement. They are attached on each other, so we can ensure to cover a particular radius for the laser measurement.

The main use for the servos is to ensure the calculated movements, at efficient battery usage and accurate movement. That means, that we have to consider certain criteria:

- How to control both servos?
- Is the battery capacity high enough to start and finish a procedure?
- Has a previous error occurred?
- Are the servo-motors ready/busy?
- Do the servo-motors stop at the calculated value?
- Have the servo-motors reached the maximum angle?

Considering the servo-control, it uses the one board-timer and the high frequency pulse width modulation (fast PWM) and the output behaviour in non-inverted mode to ensure that it can count and compare the servo values correctly.

The PWM-Signal frequency is 50Hz and the measured time is 1,8ms, with approximately 4000 steps for the configuration.

To initiate the servo delay timer, with an interrupt (output compare register(OCR)), first it has to calculate and input the capture register (ICR) value, which is used to measure the time between pulses on the external input capture pin.

Therefore, we use the following calculations:

Input comparison register:

ICR = (CPU-Frequency / 8 / 50) – 1

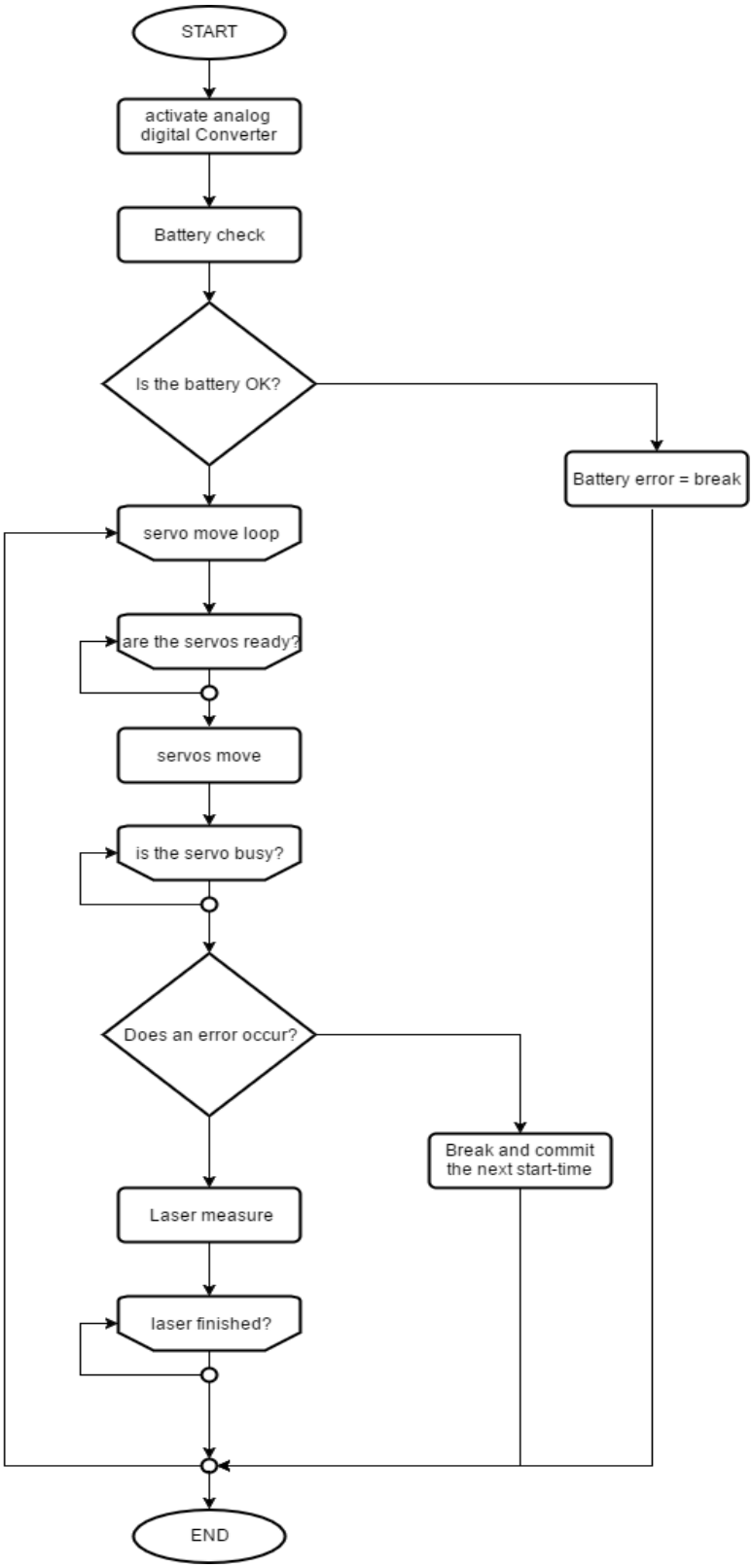
Output comparison register

OCR = (((CPU-Frequency/8/50) / 20 x 3) – 1)

After the timer is set, it can calculate the correct angles for both servos and let them drive to the measure-points. The absolute angle is 150°, because the start value can be 0° and the servos only move from 30° to 120°.

Each servo moves 20 steps per 1° (2000 steps per 1 ms). To reach 180° we need 3600 steps.

Servo-Movement-Function



Code definition

Before the move function starts, it has to ensure that the analogue digital converter for the voltage measurement in the battery check function is activated, which measures the battery capacity.
If no error occurs in the battery check function, it starts the movement function, considering that the servo is not ready and they are still moving.
As long as the servos are moving, the function holds on until they are not busy anymore, so that it can invoke the laser function, wait until the laser measurement has finished and starts the next servo movement.
If the battery check fails, the servo movement does not start, writes a statement in the error log and calls the break function.
In case an error occurs during the movement, the function checks if it is the first error and performs an additional movement, otherwise the movement error state is declared, stops the movement stops and calls the break function to commit the new start time.

Subroutines

Servo move function

In the subroutine for the servo movement, the calculation for the correct angles is defined. This function controls the current angle values, because it cannot go below 30° or over 120° (in that case the servos would drain more power).
The measure point values are defined to the output compare registers 1A, 1B and the timer starts, which ensures the movement of the servo motors.

Servo voltage measure

The function measures the power via the analogue to digital converter, to verify the that the battery capacity is high enough to complete the servo measurement.
Therefore, we sent our output data to the pin 13 (PB7) and compare both voltage values with our maximum defined voltage value.
Only if the voltage value from servo 1 and servo 2 is smaller than the maximum value, the move function will be executed.

start PWM

Starts the Timer 1, sets the prescaler and declares the public driver-state to drive.

stop PWM

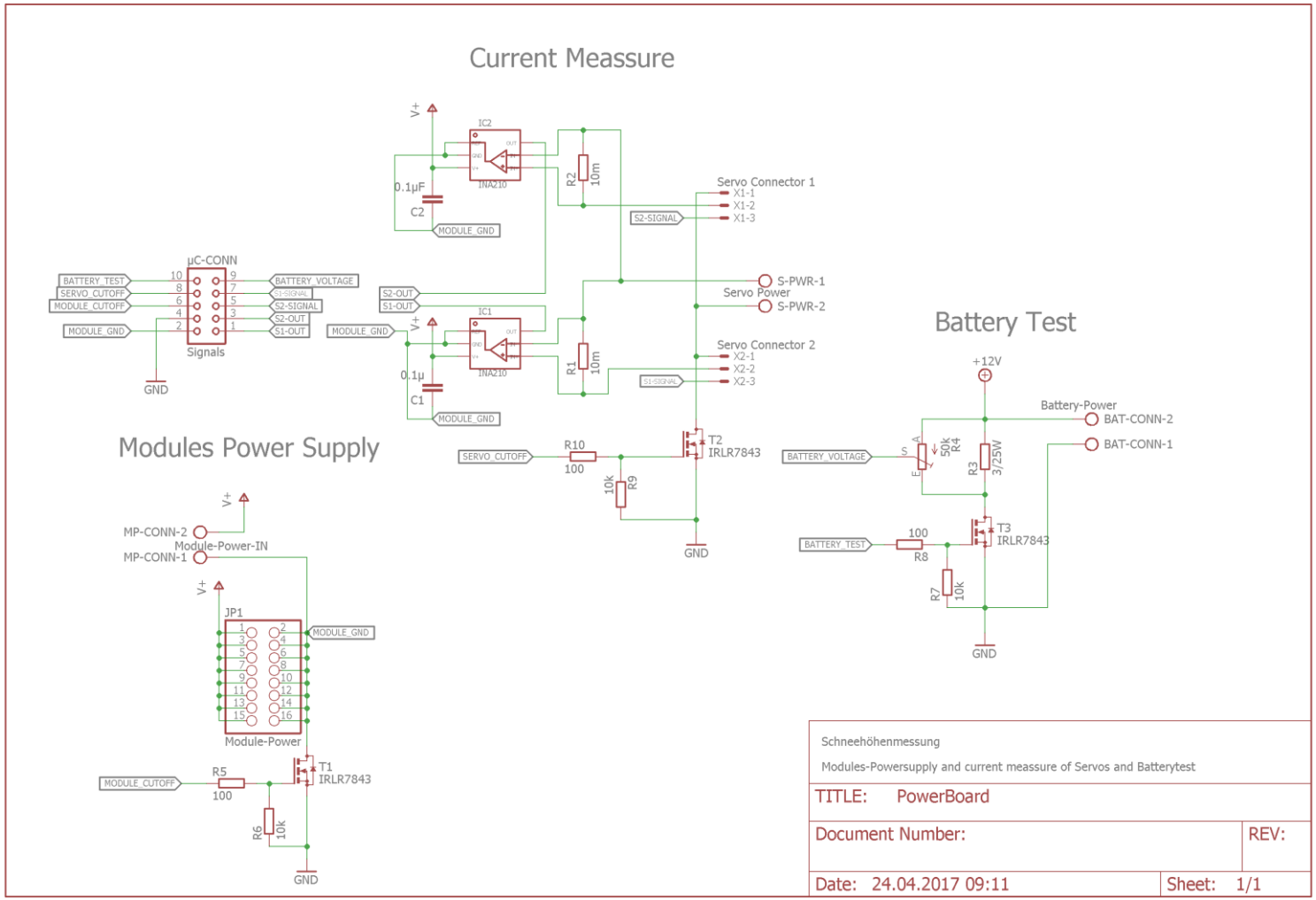
Stops the Timer 1 (deletes the bits in the prescaler register) and declares the public driver state to off.

Error log

In this function, we commit the error string to the sd-card function, depending on following error states:

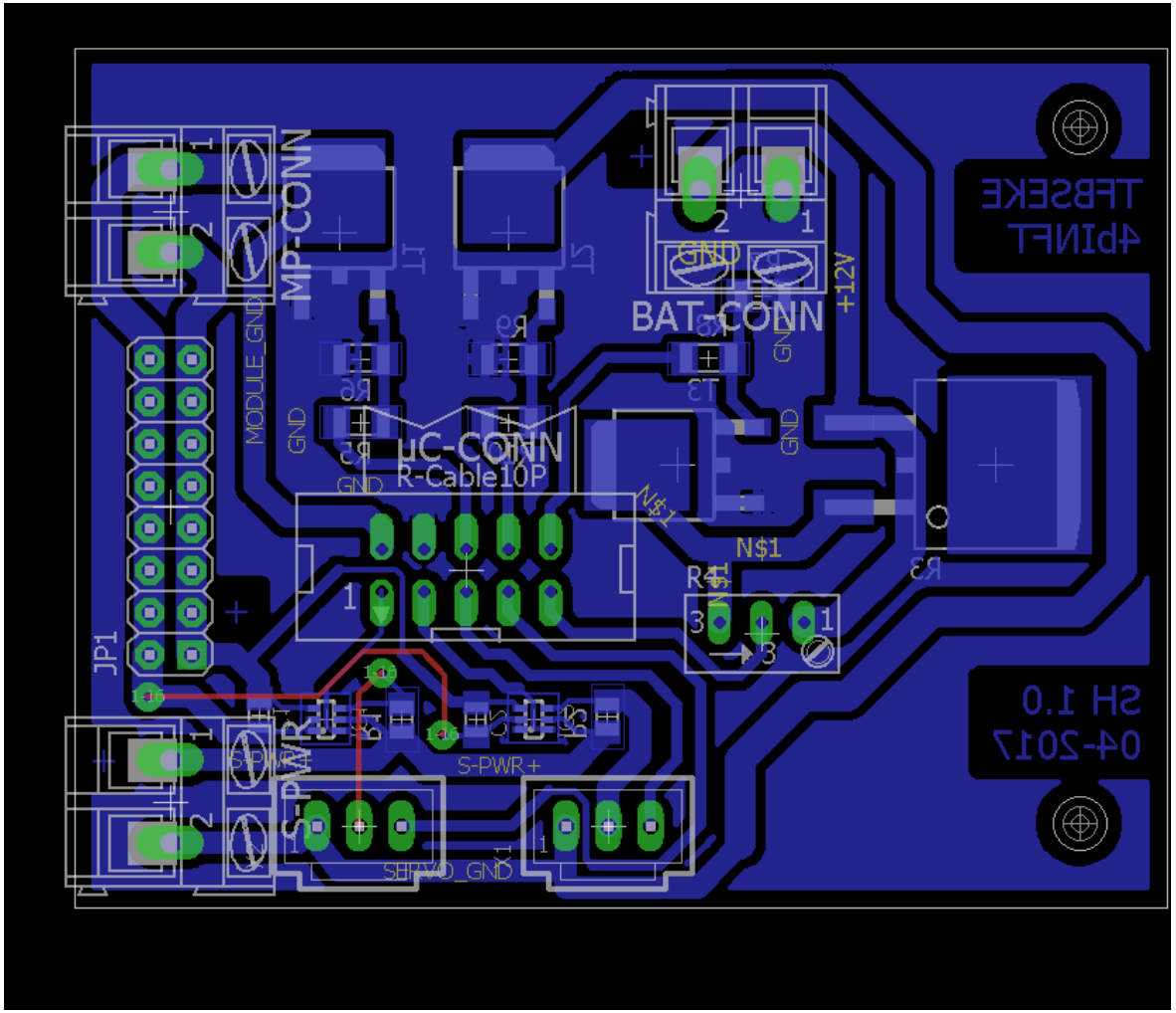
- noError no error in the function
- errorDrive error in the servos movement function
- errorBattery error in the voltage check function
- errorVoltage error in the battery check function

Servo-Control wiring



Pic. 4: Servo Control wiring

Servo-Control circuit board



Pic. 5: Servo Control circuit board

Servo Motor technical specifications (Atmel Corporation, 2014)

- MG996R High Torque Metal Gear Dual Ball Bearing Servo
- Rotation: 0 – 120 degree
- Weight: 55g
- Dimensions: 40.7 x 19.7 x 42.9 mm
- Temperature range: 0°C – 55°C
- Wire-colour:
    - Orange: PWM
    - Red: (+) Vcc
    - Brown: (-) ground

Interrupts & Timer

We will control the servo with the Pulse wide modulation control. Therefore, we have to set and configure in the following way:

- Timer (mode of operation)
- Output
    - Timer control register
    - Compare output mode (fast pwm)
- Frequency
- Duty Cycle
    - Trigger: At all times a certain value is reached, the timer will reset
- Prescaler

Interrupts

For microcontrollers interrupts will be set if:

- a predetermined time has elapsed (timer)
- a serial transmission is completed
- a measurement of the Analog-to-digital converter is completed

UART

What is UART? A UART (Universal Asynchronous Receiver(Transmitter) is the microchip with programming that controls a computer's interface to its attached serial devices.

Practical procedure:

- Set the pins on low
- Configure the pins as output
- Finding-the suitable waveforms
- Setting the mode & prescaler and start

With a UART you can easy connect an AVR with a RS-232 Interface.

Lead accumulator

First of all, why does we use a lead accumulator? Our project will be stay at the glacier. The problem is, most of the time there is a temperature of -25 centigrad. A normal battery is not able to survive. Our lead accumulator is much better for this. At -15 centigrade temperature the capacity will fall to 65%.
The internal resistance is also relevant. It is very important to know how the internal resistance is working, because any battery is losing the tension. The internal resistance can be measured with a special analyzer. The battery should be recharged every 8 weeks, which prevents sulfation.

Timer

A Timer and a Counter is the same thing.It is a chip with an integrated function module which is counting the time. The timer is also able to measure events, time intervals and periodic versions.
The counter register cannot be incremented with the wrong timer. The highest timer is the 8-Bit timer with a value of 255 steps.

Timer Overflow:

If the value is over 255 we get an overflow. That means the timer value starts again with zero. We must pay attention because we need a lot of timers and they are not too much available.

Definition:

- TCCR: Timer/Counter Control Register
- CS: clock select
- OCR: Output Compare Register

The Timer/Counter can be clocked by an internal or an external source. The source is selected by the clock select logic which is controlled by the clock select (CSn2:0) bits located in the Timer/Counter control Register B (TCCRnB).

Fast PWM

What is PWM? Pulse-width modulation (PWM), is a modulation technique used to encode a message into a pulsing signal. Although this modulation technique can be used to encode information for transmission

Definition

Provides a high frequency PWM waveform generation option. In non-inverting compare output mode, the output compare (Ocnx) is cleared on the compare match between TCNT and OCRnx and set at BOTTOM.

Lidar Lite V2

The Lidar Lite V2 Laser Rangefinder by PulsedLight is a laser based distance measurement device used to precisely measure the current snow depth.

Specs (PulsedLide Inc., 2017)

Power	4.75-5.5V DC Nominal, Maximum 6V DC
Weight	PCB 4.5 grams, Module 22 grams with optics and housing
Size	PCB 44.5 X 16.5mm
Housing	20 X 48 X 40mm
Current Consumption	<2mA @ 1Hz (shutdown between measurements), <100mA (continuous operation)
Max Range (typical conditions)	~ 40m
Accuracy	+/- 2.5cm

Measurement Technology

The distance is calculated using “signal matching patterns”, also called signal correlation. The measurement device estimates the delay between sending and receiving the light burst by comparing it to internally stored distance reference tables. This allows for very fast signal processing and overall measurement speed (50+ measurements per second are possible)

Usage in Snow depth measurement

The laser is attached to two servo motors. These motors move the laser in horizontal and vertical to measure a matrix-like area.

One single measurement is done by measuring the same point ten times and calculating the average distance. The accuracy of the distance result is improved by this procedure.

Communication between µC and Laser

The communication interface for this device is TWI (Two Wire Interface) clocked at 100 kHz.

Literature and Data

Atmel Corporation. (2014). *Atmel Atmega 2560 Datasheet.* Retrieved from http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Bosch. (2017). *BME 280 Sensor.* Retrieved from ADA Fruit: https://cdn-shop.adafruit.com/datasheets/BST-BME280_DS001-10.pdf

Datalogger. (2017). *SD Memory Reader Datasheet.* Retrieved from http://datalogger.pbworks.com/w/file/fetch/89507207/Datalogger%20-%20SD%20Memory%20Reader%20Datasheet.pdf

ElecFreaks. (2017). *Ultrasonic Ranging Module HC - SR04 Datasheet.* Retrieved from http://www.elecfreaks.com/store/hcsr04-ultrasonic-sensor-distance-measuring-module-ultra01-p-91.html

PulsedLide Inc. (2017). Retrieved from https://www.robotshop.com/media/files/pdf2/pli-03_specifications_and_hardware.pdf