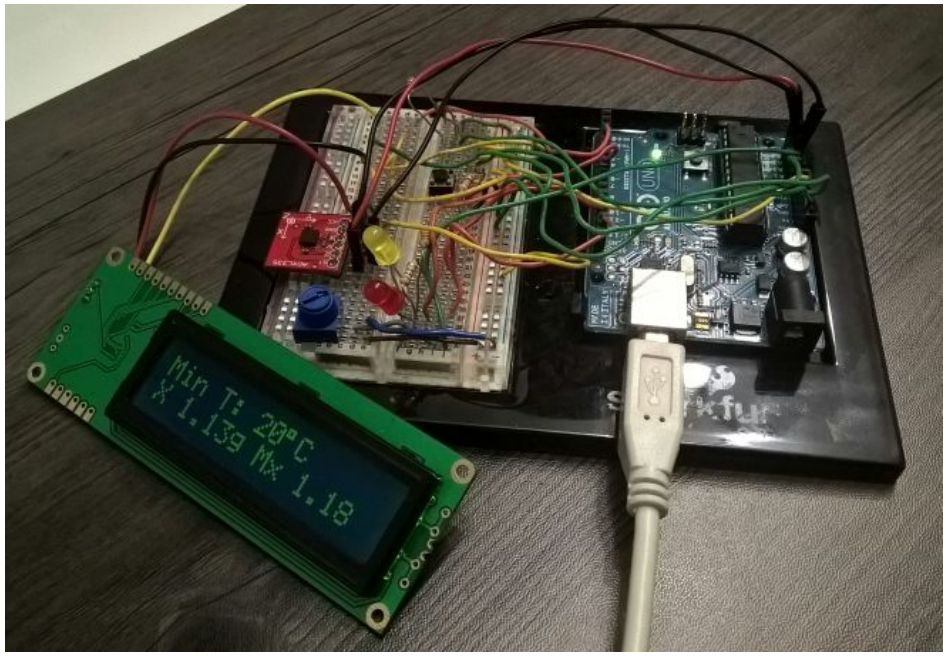
Project 1 - Smart Sensor

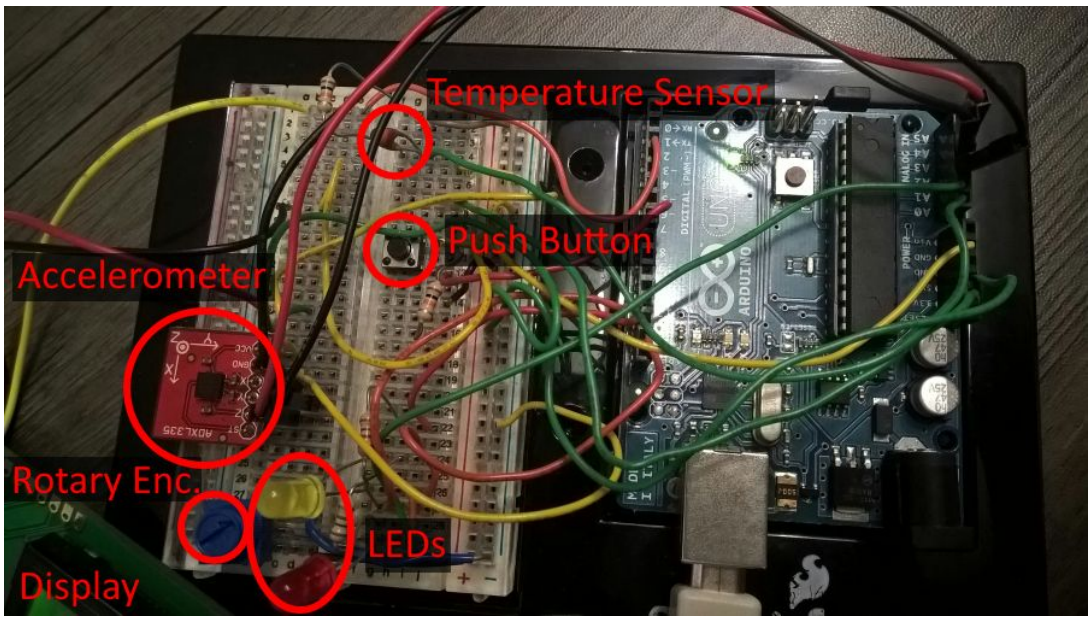
Niklas Hösl, Magdalena Radinger

Source: https://github.com/hoenic07/sensors-networks-projects

Pictures of Arduino setup



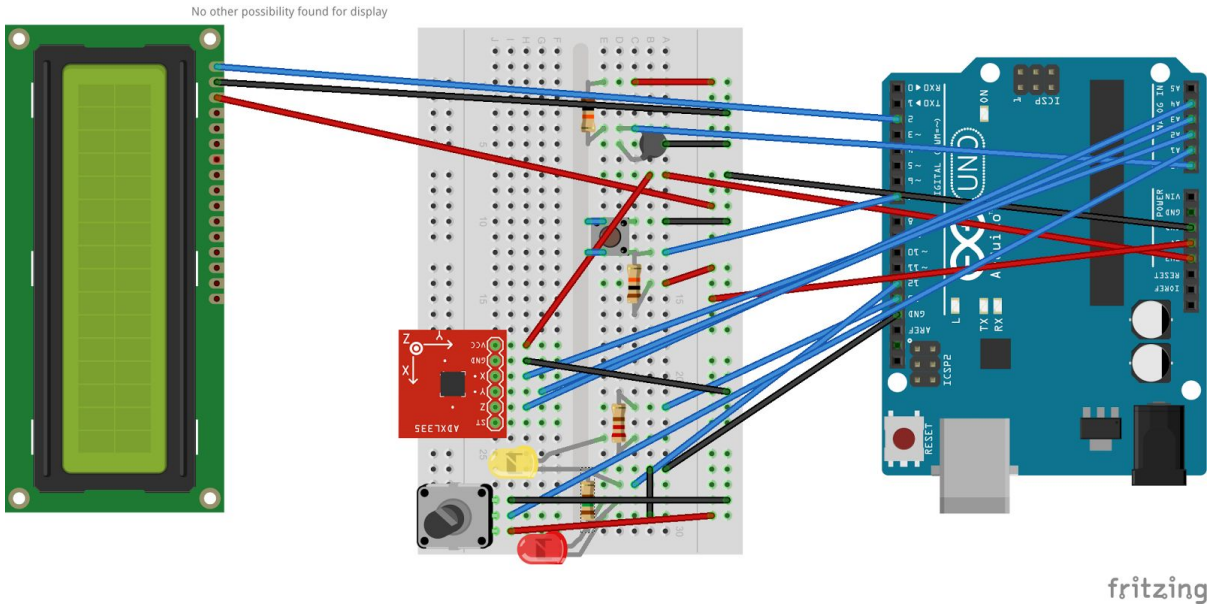
General Structure



Important Components

Electric Circuit

Note: We couldn't find the correct component for the display. The ports are normally at the back of the display.



Cable color scheme: Blue: Data, Red: VCC, Black: GND

Protocol

A message has the following structure:

START	LEN	RECEIVER	COMMAND	DATA_FORMAT	DATA	END

Name	Description	Length (Byte)
START	Start byte (0x55)	1
LEN	Total length of the message in byte	1
RECEIVER	Id of the Receiver (0...bus(client), 1...sensor)	1
COMMAND	Command to execute	1
DATA_FORMAT	Format of the user data.	1
DATA	User data	0-99
END	Stop byte (0xAA)	1

Commands

Command	Direction	Description	Value	Data Format
REQ_ACK	->sensor	Request ACK e.g. to check connection	0	EMPTY
REQ_TEMP	->sensor	Request current temperature value	1	EMPTY
RESP_TEMP	->bus	Response with current temperature value	2	VAL
REQ_X(_Y,_Z)	->sensor	Request current acceleration value (per axis)	3-5	EMPTY
RESP_X(_Y,_Z)	->bus	Response with acceleration value (per axis)	9-11	VAL
REQ_PITCH(_ROLL,_THETA)	->sensor	Request pitch/roll/theta value	6-8	EMPTY
RESP_PITCH(_ROLL,_THETA)	->bus	Response with pitch/roll/theta value	12-14	VAL
REQ_PARA	->sensor	Request a certain parameter	15	PARAM
RESP_PARA	->bus	Response with parameter value	16	PARAM_VAL
SET_PARA	->sensor	Change a parameter value	17	PARAM_VAL
ALARM_TEMP_DT	->bus	temperature change in time value exceeded	20	EMPTY
ALARM_TEMP_MIN(_MAX)	->bus	temperature min/max value exceeded	18-19	EMPTY
ALARM_ACC	->bus	accelerometer threshold value exceeded	21	EMPTY
RESET_MINMAX	->sensor	Reset all minimum/maximum storages	22	EMPTY
ACK	->bus	Success confirmation	23	EMPTY
NACK	->bus	error	24	EMPTY

Data Format

Data Format	**Data**	**Length of Data (Bytes)**
EMPTY = 0	-	0
VAL = 1	2-byte signed integer = original value *100. Eg. value to be sent: 12.345 Will be converted to: 1234 Byte[0]: 4 (0000 0100) Byte[1]: 210 (1101 0010)	2
PARAM = 2	2-byte signed integer Byte[0]: 0 Byte[1]: Parameter value (see table below)	2
PARAM_VAL = 3	2 2-byte signed integers: First short: see PARAM, Second short: see VAL	4

Parameter

Parameter	Description	ID	Default Values
(LOWER_)UPPER_TEMP_THRESHOLD	Lower/Upper threshold value for temperature	0-1	10-30°C
TEMP_PER_TIME_THRESHOLD	Threshold value for temperature change in time	2	5°C/10s
TOTAL_ACC_THRESHOLD	Threshold value for total acceleration change	3	3g
(MAX_)MIN_TEMP	Min/Max measured temperature value	4-5	20°C
MAX_ACC_X(_Y,_Z)	Max value for acceleration value (per axis)	6-8	0g
TEMP_UPDATE_INTERVAL	Interval of temperature value update	9	60s

TEMP_UPDATE_DELTA	Update delta of temperature value	10	3°C
CALIBRATION_TEMP	Calibration value for temperature	11	0°C
CALIBRATION_MIN_X(_Y,_Z)	Calibration min value for acceleration (per axis)	12,14, 16	-1g
CALIBRATION_MAX_X(_Y,_Z)	Calibration max value for acceleration (per axis)	13,15, 17	1g

General Notes:

- Little Endian is used for data transmission
- When BUS sends message to SENSOR, SENSOR always has to sent a response message. If there is no obvious response (e.g RESP_X will always follow REQ_X, but RESET_MINMAX has nothing like that) the sensor needs to respond with ACK.
- When SENSOR sends MESSAGE to BUS there is never a response message

Sensors

Temperature

Temperature values

To convert the measured voltage values of the temperature sensor to “real” temperature values we linearized the values based on a pre-calculated table. The values are then interpolated according to the values in the table.
We built a table from 1-1023V with steps of 50V (50V, 100V, 150V,...).
Steps to calculate the temperature:

1. Determine how much worth are x steps (Volt) considering the supply voltage (U = 5V)
   Ux = x * 5 / 1024
2. Determine Rx using the reference value from the data sheet.
   Rx = 10000 / (5 - Ux) * Ux
3. Calculate the temperature in Kelvin we used the provided formula and values (A-D) of the data sheet.
   T(Rx) = 1 / (A + B * ln(R /Rref) + C * ln(R/Rref)^2 + D * ln(R/Rref)^3)
   Where
   - A = 0,003354016
   - B = 0,0002744032
   - C = 0,000003666944
   - D = 0,000000137492
   - Rref = 22000
4. To get the Temperature in degrees (°C) just subtract 273,15.

This is the result of the calculations:

x	Ux	Rx	T (°C)
1	0,004883	9,775171	445,6739
50	0,244141	513,347	149,2655
100	0,488281	1082,251	117,9297
150	0,732422	1716,247	100,5953
200	0,976563	2427,184	88,44942
250	1,220703	3229,974	78,95258
300	1,464844	4143,646	71,02608
350	1,708984	5192,878	64,10961
400	1,953125	6410,256	57,87106
450	2,197266	7839,721	52,09248
500	2,441406	9541,985	46,61727
550	2,685547	11603,38	41,3221
600	2,929688	14150,94	36,09953
650	3,173828	17379,68	30,8448
700	3,417969	21604,94	25,4426
750	3,662109	27372,26	19,749
800	3,90625	35714,29	13,56015
850	4,150391	48850,57	6,54546
900	4,394531	72580,65	-1,92807
950	4,638672	128378,4	-13,424
1000	4,882813	416666,7	-34,8462
1023	4,995117	10230000	-81,1968

Calibration of temperature values
If you set the calibration value to 23°C and the current temperature value is 22.4°C the difference (-0.6°C) is stored and always added to the currently measured temperature value.
Diff = measuredTemp - calibrationValue
Temp = measuredTemp - diff

Accelerometer

Getting acceleration values

The values that come from the analog port are theoretically from 0 to 1023. These must now be mapped to accelerations in g.
The 0g value is around 512. 1g can be found at around 357 and -1g is around 667.
An incoming voltage value will than be mapped according to these values.

Tilt

Pitch, Roll and Theta are calculated according to following formula:

$$\rho = \tan^{-1}\left(\frac{A_x}{\sqrt{A_y^{\,2} + A_z^{\,2}}}\right) \quad \Phi = \tan^{-1}\left(\frac{A_y}{\sqrt{A_x^{\,2} + A_z^{\,2}}}\right) \quad \Theta = \tan^{-1}\left(\frac{\sqrt{A_x^{\,2} + A_y^{\,2}}}{A_z}\right)$$

The resulting values are in radiant.
To get degrees these values are multiplied by 57.3 (~ 180 / PI)

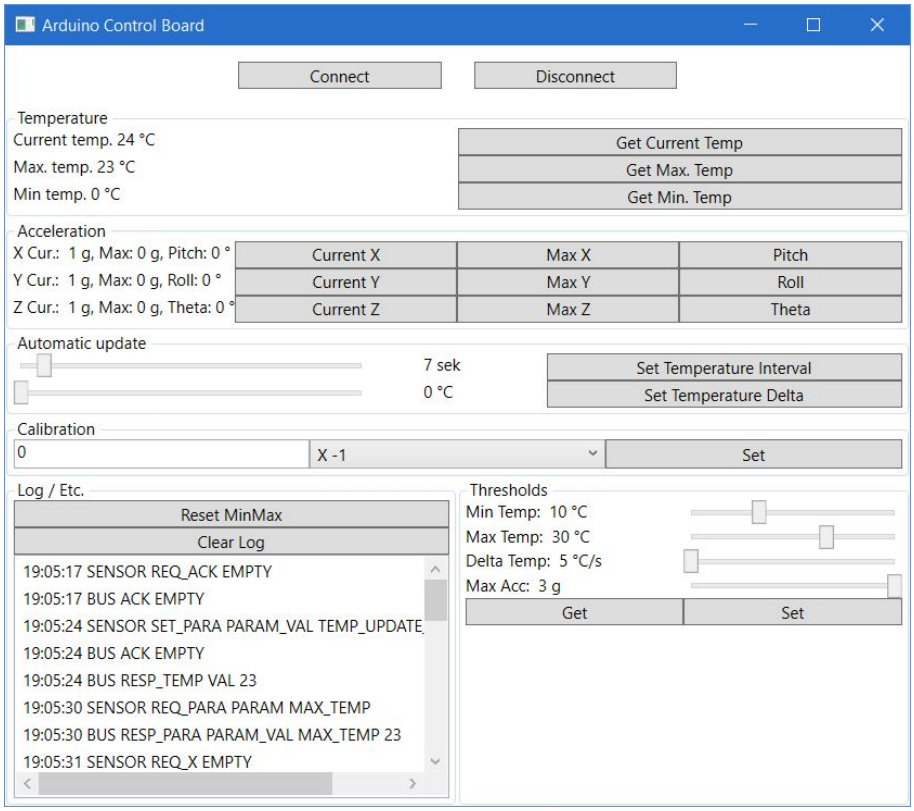
Calibration of accelerometer values

The client can send min/max calibration accelerations for each axis. Such a value could be e.g. 1.10g for MAX Y.
This value will than be used instead of 1g for the value mapping. 1.1g is than the new value that would be output for 357.

Client

The client is a WPF Windows Application written in C#.
It connects to the COM3 Serial Port of the PC.

Screenshot



User Guide

Connect the Arduino via a USB-Cable with the PC. Start the client and press the Connect Button. After a successful connection a message box “Connected” pops up. Otherwise an error message occurs.
After that all commands become active. The UI is kept as simple and intuitive as possible sp that no further description should be needed.

Note to the automatic update: For both sliders the value 0 indicates no automatic update.

Class diagram Arduino

The following picture shows the class diagram of the Arduino project. The **red** one is the main of the project which contains the setup() and the loop() methods. The **green** boxes represent sensors and input devices where “Input” is the rotary encoder and the push button. The **blue** boxes are output classes to display the results/alarms. The **gray** boxes represent management classes and the Parameters class (**yellow**) holds all the default values, min/max values which were set by the client or measured by the Arduino.