

# Smart RFDuino

Alexander Baldamus

Dominik Schlütter

Hinrikus Wolf

Sommersemester 2014



# 1 Idea

A façade as “thinking skin”, wrapped around a building—this doesn’t have to be a touchable, physical membrane. It might also be a virtual skin, an invisible network covering an architectural entity.

We started with the idea of an “augmented reality window” to collect and display data gathered from other modules in one convenient location. But after some tests and prototypes, we had to redefine our strategy: not just showing things, but to enable active collaboration between the different elements developed for the “multimodal media madness” lab at the Media Computing Group (RWTH Aachen University).

So we started to look into collecting data with an intelligent sensor and actuator network consisting of multiple loosely coupled radio transmitters. Our first goal was to create services for other façade elements to react on different events being broadcasted, not depending on a specific sensor module. The next step was creating a use case that transcends the use of sensor elements as simple data sources for the actual façade and enables direct interaction with the architectural environment. For example you could take the light switch from a wall nearby and bring it with you to your desk, so you could direct the lighting from the place where you need it—and where it is much easier to assess.

But as there’s no fixed 1:1 relationship from sensor to actuator, there are still other services and sensor readings that might be relevant, e.g. a PIR movement sensor, overall brightness value, etc.

We chose the Arduino environment to create our evaluation nodes, using ubiquitously available AVR ATmega Chips and HopeRF radio modules. This cost efficient hardware makes it easier to achieve a closed-meshed sensor network—our “thinking skin”.

You can find all necessary documents for the hard- and software in our Repository 7.

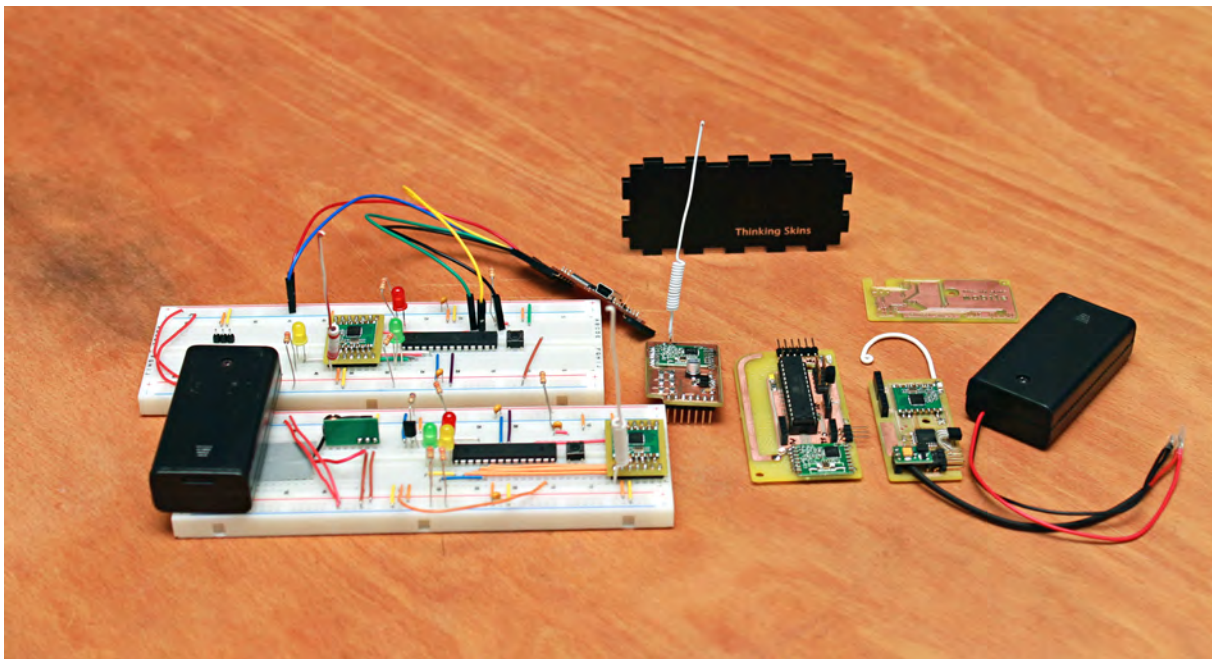


Figure 1: Collection of our build RFduinos

## 2 Radio Technology

After extensive research we decided to use *RFM69* radio modules by HopeRF, based on the Semtech SX1231 transceiver. They are quite powerful, reasonable simple to interface with (SPI) and there is already a well tested Arduino library [7].

The main features are:

- 256 networks with max. 255 nodes each
- customizable transmit power
- interrupt driven, sleep functions
- hardware AES encryption

### 2.1 Antenna

Another field for experimentation is antenna technology. The most basic antenna consists of a straight piece of wire with the length  $\lambda$  of the transmitting wavelength. At 433MHz (that's what we use) this would be about 70cm, which is rather unwieldy. It is possible to shorten the antenna to  $\lambda/4$  without too much loss, bringing the size down to about 17.5cm—better, but still rather large.

According to [2] this can be brought down to about 8cm. With these hand wrapped coil antennas we were able to get impressive ranges of 300m to 500m. The same idea (using a coil or inductance) makes it feasible to use PCB antennas with 433MHz radio modules.[3]

### 2.2 Sensor Networks

There are various ideas and designs for building sensor networks that communicate via inexpensive radio modules. The most notable examples with regard to our project are [4] and [5], which also use HopeRF transceivers. On the hardware side they focus on providing an Arduino compatible base module to extend with separate shields while keeping most of the Arduino feature set, but thanks to the Arduino environment we could benefit a lot from their work with the RFM69 libraries.

We tried to focus on a small footprint sensor platform with few key connections like OneWire and I2C. To connect to other Arduino boards we implemented a variant of the [6] to be built on a breadboard, interfacing with the standard 5V logic.

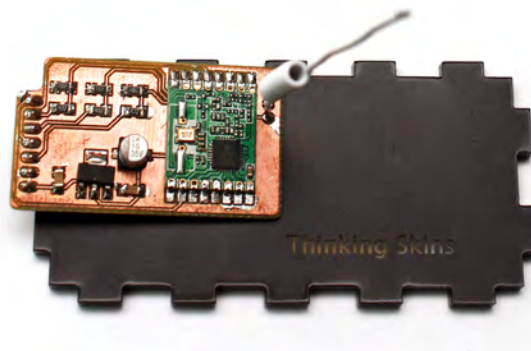


Figure 2: RFM69 Breakout Board (BOB)

### 3 Mobile RFDuino

Our main concern was low energy consumption, simple hardware design and flexible use of board sizes—so we built our own custom arduino-compatible boards. These design requirements led to a simplified bare bones Arduino, using the internal oscillator as a clock source and running on 3.3V (which means less power draw and easier integration with the RF modules).

#### 3.1 Generic Mobile Sensor

Starting with a bare bones “Arduino on a breadboard” (Figure 3), we tried to minimize the board size while still keeping access to all the pins on the *ATMega328P*. The board is powered by two AA batteries, converted to 3.3V via an *NCP1402* boost converter (and an additional sensor to monitor the battery voltage).

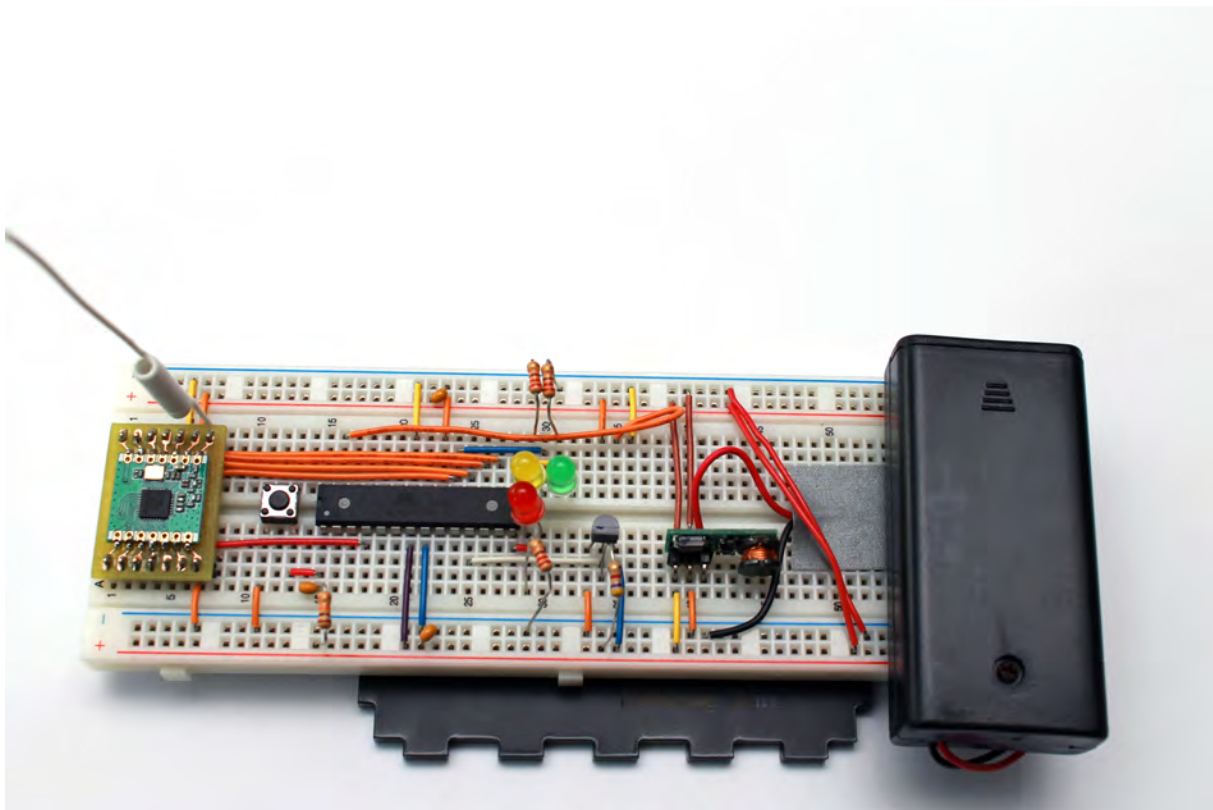


Figure 3: “Arduino on a breadboard”

The Generic Mobile Sensor consists of an *ATMEGA328P-PU*, a *RFM69W* radio, two LEDs (pin 8 + 13), a PCB antenna and connection pins for the ICSP programmer, serial programming (standard FTDI connector) as well as sensor inputs via I2C and OneWire (2 times, accessible on pin 3 + 4). The other pins are grouped in a dedicated expansion port to allow easy customized extension boards, as shown in Figure 4.

By using a 56nH inductor, we were able to shorten antenna length on the board to 70mm (this is described in the app notes for our radio modules). In combination with 1203 SMD components, this results in a small package of just 72 mm x 38 mm.

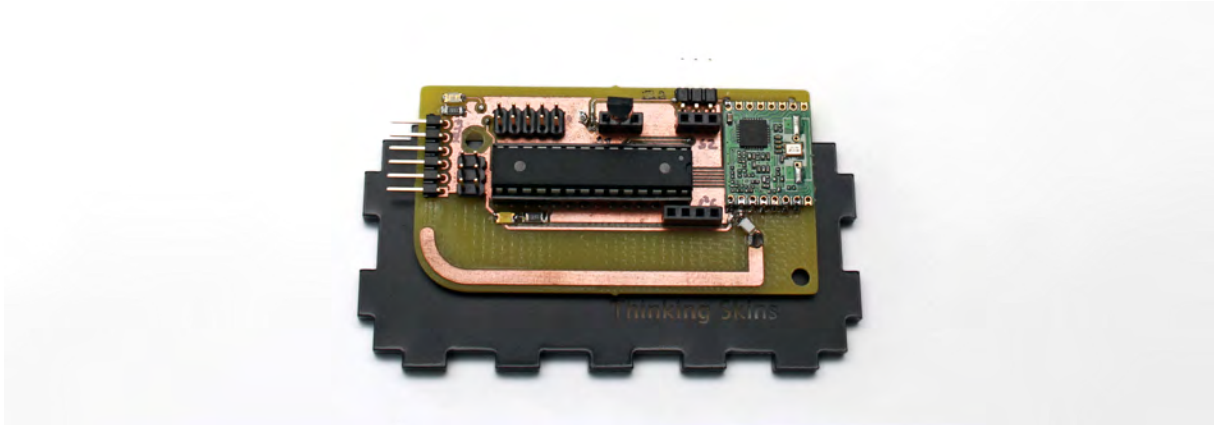


Figure 4: The Generic Mobile Sensor Platform

### 3.2 Mini Mobile Sensor

In an attempt to further reduce the board footprint, we redesigned it again—this time using a *ATMega168-AU* (in TQFP SMD package), the smaller *RFM69CW* and 603 SMD components. The LED on pin 8 and a *DS18B20* digital temperature sensor are soldered to the board, additional OneWire and I2C connectors are provided. Programming is done with the standard 6 pin ICSP connector (angled 90deg for minimal height).

The short antenna is not on the PCB this time, giving more flexibility for the case. These efforts result in a tiny board 57mm x 27 mm, less than 60% of the Generic Mobile Sensor. This size fits on top of a double AAA battery case, although this would strongly affect the time between battery changes (eneloop: 800mAh for AAA vs. 2100mAh for AA) and can be seen in Figure 5.

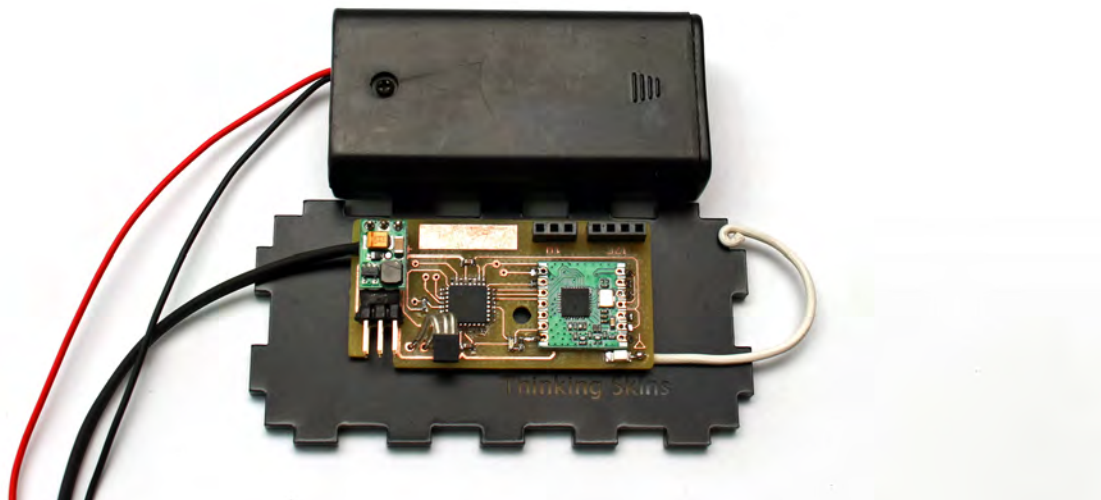


Figure 5: The Mini Mobile Sensor

### 3.3 Rotary Encoder

One of our first ideas to build a light switch, which you can take from the wall and control lights e.g. from your desk. Especially we took instead of a normal switch a Spark Fun [8] rotary encoder with an LED bar

graph around it.

We combined this module with an Arduino and one of our radio modules. The encoder gives us a value between 0 and 63 we can send to another Arduino and encode this e.g. as a PWM signal, for this we built a case. See in Figure 6 and on the title page. In the case is an Arduino Uno with on of our half shields (see Section 4) and a 9V battery.



Figure 6: Rotary Encoder

## 4 Arduino Half Shield

We designed a half shield for the Arduino platform. It contains a voltage divider to interface the *RFM69W* module (running on 3.3 V) with the standard 5V Arduino Board. This allows easy prototyping without building custom ATmega boards.

The half shield covers the left side of the Arduino, using the power connections on the lower half and the SPI interface pins on the upper half. The unused I2C pins on the Arduino UNO r3 layout are made accessible on the shield itself, together with +5V and GND in a convenient pin header - ready to plug in additional sensors. The power for the RF module is drawn from the 3.3V pin of the Arduino, supplying up to 50mA. This is enough for the default *RFM69W* with +13dB (using max. 45mA), the more powerful *RFWM69HW* (+20dB) would need an additional voltage regulator.

It also features our default LED on pin 8 to signal RF activity and a removable jumper to either use pin 9 as the interrupt pin or to connect the Arduino hard wired interrupt on pin 2. And of course there's the *RFM69* module as well as a removable antenna (and a matching ground plane on the backside of the half shield).

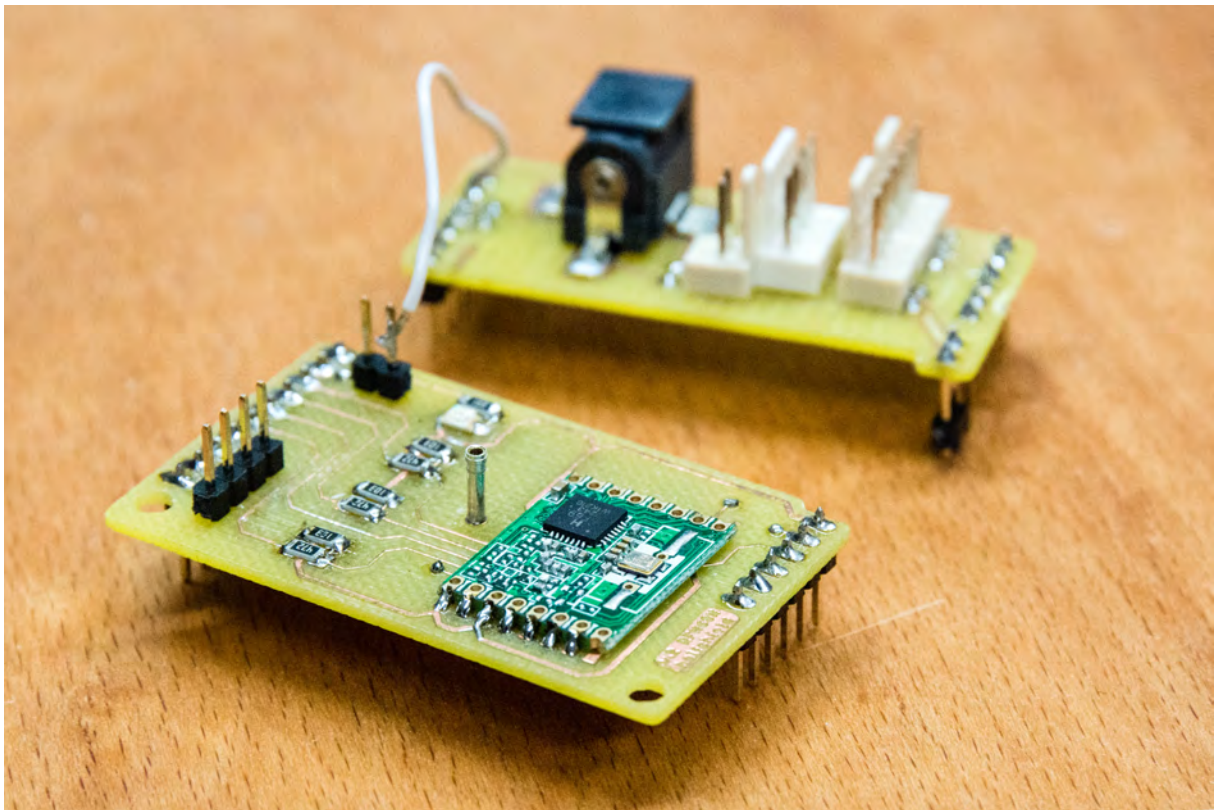


Figure 7: RFM69 half shield, tied to another module for façade control

## 5 Software

The low level communication uses the protocol of the RFM69 library, this leaves us with a maximum of 61 bytes to implement our higher level protocol for sensor communication.

Every message starts with the concatenation of #Class#ID#Position#Data#CustomString#. We have reserved two bytes for each of those common attributes, except for position which only contains 1 Byte. We distinguish between predefined and custom classes as well as acknowledged messages and unacknowledged messages, so we split the class-address room. The classes 0x0000 - 0x00FF are predefined classes for unacknowledged messages. Classes 0xFF00 - 0xFFFF are the same classes but request an acknowledge when sending. All other classes are custom, with 0x0100 - 0x7FFF unacknowledged and 0x8000 - 0xFEFF acknowledged messages.

The attribute ID is a unique ID for each module in this class. The attribute Position may depend on the class, but 0x00 means this module is portable, so it has no fixed position. In the attribute CustomString we can decode additional data, if needed.

Here is a list of our predefined classes.

- ACK\_TEMP = 0xFF01
- TEMP = 0x0001
- ACK\_SOUND = 0xFF02
- SOUND = 0x0002
- ACK\_ROTARY = 0xFF03
- ROTARY = 0x0003
- ACK\_BATTERY = 0xFF04
- BATTERY = 0x0004
- ACK\_MOTION = 0xFF05
- MOTION = 0x0005

The main part of our software is designed to make sending and receiving sensor data accessible and easy. Therefore we provide functions to, for example, set the class or the sensor data for the next sending action. The software takes care of necessary bit-shiftings and actions to store all data in one byte array, ready to be sent via the RFM69 library. The software also handles the other way round, receiving data over RFM69 and providing direct access to the data such as class, position and sensor-data. The only thing to be taken care of is polling the receive function repeatedly to not miss any sendings. We also provide the possibility to send float as well as long type sensor data. As we do not send information about datatypes, the receiving end needs to know, whether to interpret the data as float or as long. The user relevant methods to be used are:

- sendRFM()
- sendRFM(word gateway, word klasse, word id, byte pos, long value, byte custom[], int customSize)
- sendRFM(word gateway, word klasse, word id, byte pos, float value, byte custom[], int customSize)
- setNextSend(word klasse, word id, byte pos, long value, byte custom[], int customLength)
- setNextSend(word klasse, word id, byte pos, float value,



- ```
byte custom[], int customLength)
```
- `setNextSend(word klasse, word id, byte pos, long value)`
  - `setNextSend(word klasse, word id, byte pos, float value)`
  - `setNextKlasse(word value)`
  - `setNextId(word value)`
  - `setNextPos(byte value)`
  - `setNextWert(word value)`
  - `setNextWert(float value)`
  - `setNextCoustom( byte* array, int length)`
  - `receiveRFM()`
  - `getKlasse()`
  - `getPos()`
  - `getId()`
  - `getWert()`
  - `getFloatWert()`

We also programmed a standard code for most of our sensors. With defines, it's possible to determine if, for example, the mobile sensor has a temperature sensor on it, how often the value should be sent and if it should suspend itself after sending data to save energy. With these standard definitions, it is possible to provide our code quick and easy for several different sensor types.

Please note that if you try our code from the Bitbucket repository, you may need to download several libraries such as OneWire or the RFM69 library. You also need to place our m3RFM.h library in the Arduino libraries folder.

## 6 Interaction

Due to the nature of our project, inter-group collaboration was an important factor for our work—on the hardware as well as the software side. The Fab Lab Aachen[1] was our main hub not only to exchange ideas and connections with other groups, but also as a workspace for our own experiments and resource for missing components.

For the hardware we had different approaches. One group had developed their own Arduino shield, although they were using a different set of pins. We settled on building two half shields (one using the left and one using the right half of the Arduino). The other variant was integration of our radio components with the existing schematics together on one single Arduino shield. In this case we also had to implement a 3.3V voltage regulator to satisfy the power needs of both projects. For the remote sensors we tried to keep the footprint as small as possible in order to minimize interference with the rest of the architectural design. We use simple AA rechargeable batteries for our prototypes—that's what limits our goals of minimization, custom lithium batteries might keep the sensor nodes even smaller. In order to connect the projects of different groups, we provided different system types.

### 6.1 Mini Mobile Sensor

Using our own library, we build two sensor-systems with temperature sensors and the possibility to connect additional sensors via OneWire or I2C (cp. figure 5). Both automatically send the temperature and then power down to save energy, as they are battery driven. After 8 seconds, they power up again, measure the temperature and send it out. The temperature data are received and processed by another group, who also use our library to access the data. Additionally, every 20 loops, they measure the voltage of the attached battery and broadcast it to every node on their network. This way, one are able to receive this broadcast and check the status of the mini mobile sensors battery.

### 6.2 Generic Mobile Sensor

An earlier version of our mini mobile sensor, the generic mobile sensor (cp. figure 4), provides an additional analog data pin. We use it to measure the temperature, check for people nearby the connected PIR-sensor and check the volume-level of sounds nearby. All these data are send to the same group as the mini mobile sensors, but the sound-level is dispatched each 25 milli seconds while the other sensor data are dispatched each 10 seconds.

### 6.3 Rotary Encoder

As described in section 3.3 (cp. figure 6), we build a rotary encoder with the idea to control the light level of a room with a portable switch. As another group planned to build a window, which could darken the room if it is to bright, the idea came up to manually control it with our rotary encoder, if needed. Therefore we provided a shield with a RFM69 module as well as our library. The rotary encoder can be rotated, filling up a bar graph to show the level of rotation. If the encoder is pressed, it sends out the actual value to the window which processes it. To gain easy access to the data, the other group again used our library. The rotary encoder system is able to track any rotation at the encoder, make it visible at the bar graph and send it out after acknowledgement. If a change in rotation is not acknowledged after 5 seconds,

the rotary encoder drops the rotation and change its state back to the old. In this case, the value is not sent to the window. The encoder system itself contains an Arduino UNO, a RFM69 half shield and the encoder. Because the encoder is, as well as the RFM69 modules, interrupt driven and needs to be highly interactive, we needed a different code as our standard mobile sensor code. The standard code would not work on the rotary encoder as we needed 3 interrupt pins, but the Arduino UNO only provides 2 hardware interrupts. The rotary encoder also needs many different pins, so that a complete different code was the easy solution rather than blowing our standard code up too much.

## 7 Links

<https://bitbucket.org/ForrestGun/m3-public/src>

## References

- [1] Fab Lab Aachen <http://hci.rwth-aachen.de/fablab>
- [2] Shortened 433MHz antenna <http://www.elektor.nl/Uploads/Forum/Posts/How-to-make-a-Air-Cooled-433MHz-antenna.pdf>
- [3] HopeRF: Antennas Applications for RF module [http://www.hoperf.com/upload/rf/ANTENNAS\\_MODULE.pdf](http://www.hoperf.com/upload/rf/ANTENNAS_MODULE.pdf)
- [4] JC's Environmental Electronics <http://jeelabs.org>
- [5] LowPowerLab Moteino <http://lowpowerlab.com/moteino>
- [6] PCB Design of radio transceiver breakout board [https://github.com/uChip/RFM69W\\_BOB](https://github.com/uChip/RFM69W_BOB)
- [7] RFM69 library for Arduino <https://github.com/LowPowerLab/RFM69>
- [8] Spark Fun Rotary Encoder Breakout <https://www.sparkfun.com/products/11040>