Internet of Things – Project Work

Track.ai

Philipp Kast
Module MTE7117, MSE, Master of Science in Engineering, BFH

Use case

I want to investigate the use case of tracking objects in the real world and the analysis of this data by using machine learning approaches.

The long-term goal would be to automatically detect delays, missing objects, cost savings through optimizations and error prevention.

First and foremost, I want to investigate the Internet of Things part, which means to build and program a tracking device, which can send out the position to the cloud. The position sent to the cloud has in a second step to be stored and displayed. A stretch goal would be to analyze the position data by machine learning approaches, as an example within BIM (Building Information Modelling) for tracking objects from the factory to the construction sites.

Sensors

There are different approaches for positioning, the most well-known is positioning over GPS, as we have this technology in our mobile phones. Of course, this today goes along with what is called assisted GPS, which augments GPS by using cell tower data to enhance quality and precision when in poor satellite signal conditions. That said, positioning can also happen over the cell tower data of the mobile telecommunication network. Apart from that, other approaches exist, such as positioning beacons, or positioning over light sensors as applied for bird tracking.

For our application, obviously a certain accuracy is needed, for most applications as accurate as a couple of meters, for others a couple hundred meters would be good enough. For the first we need GPS, for the latter positioning over cell tower data would be good enough.

Network

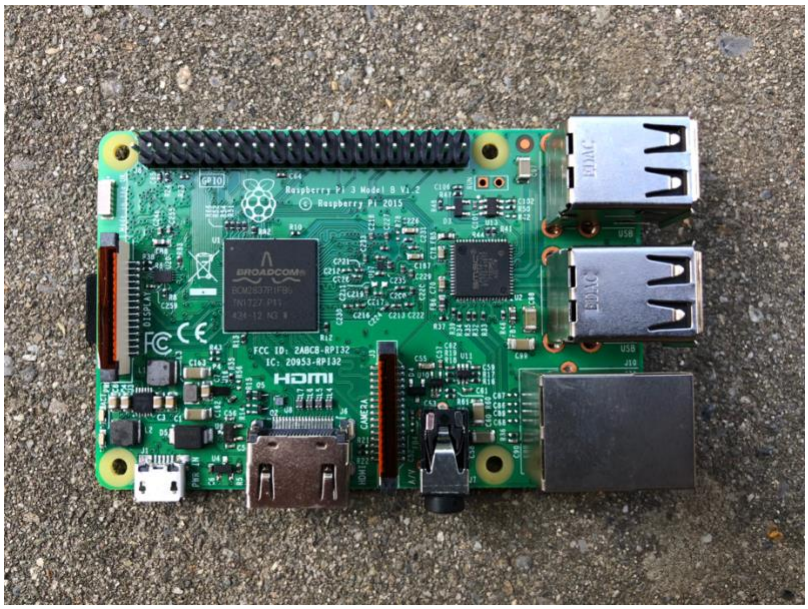
For tracking we obviously need a very good network coverage, as we want to track objects everywhere. For our use case the positioning is not limited to a small geographic area, which excludes most network technologies. For our use case the mobile telecommunication network is the only realistic choice.

Device

For tracking purposes different devices are already on the market. For certain tracking applications power consumption is not a problem, e. g. in cars or trucks, where you have access to electric power. Some battery powered trackers have today already pretty long battery life times, up to something around 10 days max (realistically though is a couple of days, 2 to 3 days). This is due to the fact, that these devices operate over GPS and over the mobile telecommunication network, both of these technologies are not low power. For our use cases this is not really good enough. You can charge a tracker for your pet every other day, but you can’t do that if you have trackers out there in a large number, e. g. on parts for a construction site.

There are multiple technologies designed for low power networking. A prominent one, with a large signal range of up to multiple kilometers, is LoRa. There are devices like the LoRa Feathers, which let you create flexible wireless networks. But you always need a gateway to the internet. This can be easily done by installing a device within reach, which has access to the internet and acts as a gateway. This can be done by using a Raspberry Pi, which you can connect to the internet and connect to devices over LoRa.

I experimented with the Feather M0 above, which works well for data amounts necessary to transmit position data.

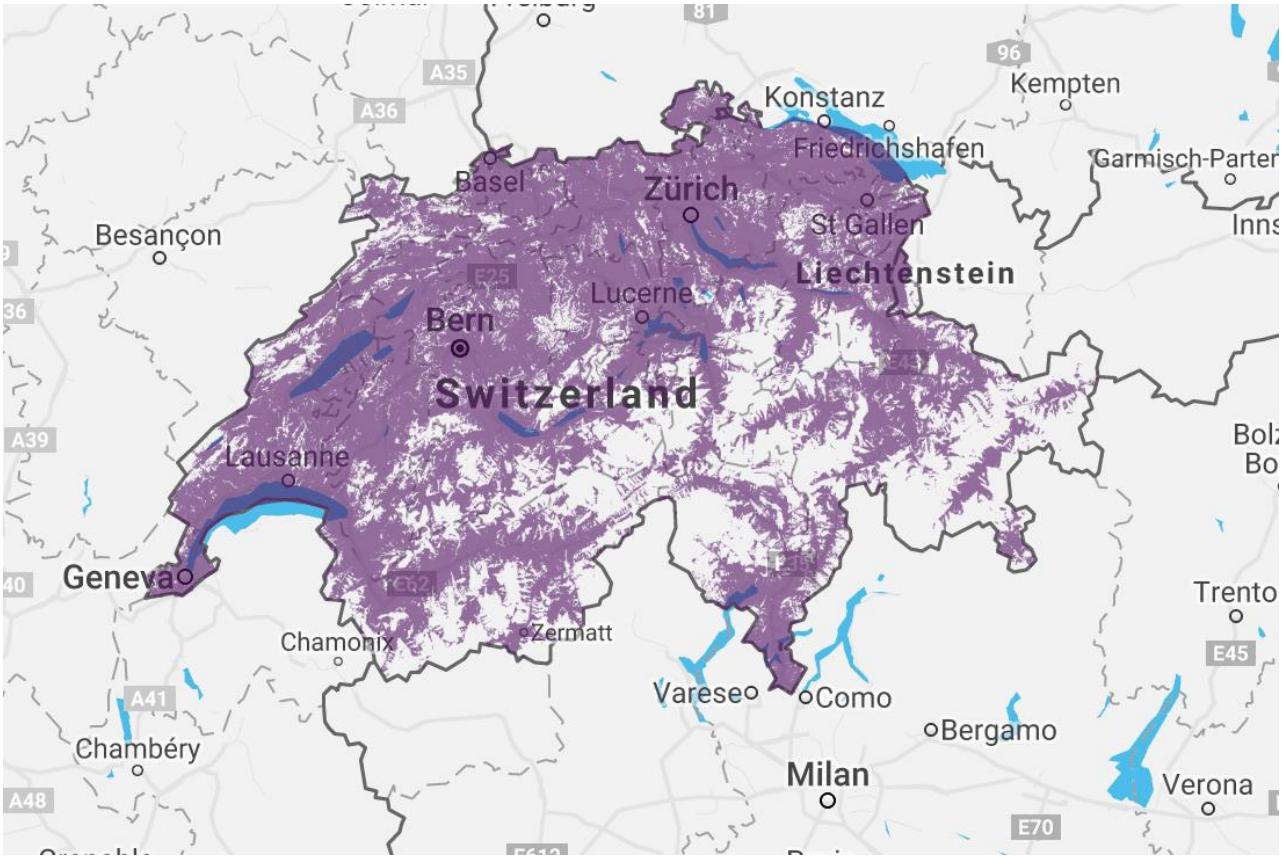


A Raspberry PI 3 can be connected to a Raspberry Pi 3 to the internet over WLAN (or an Ethernet cable), but you obviously need to have a WLAN network and access to it. But if you have WLAN, you have already your IP gateway to the internet.

For our case this approach has an obvious issue, we can't build up a network of internet gateways for the whole area of interest. While this would work nicely for a construction site, we would like

to track objects not only on the construction site itself, but we would like to track objects from the factory, which could be very well even far outside of Switzerland.

Prominent mobile telecommunication companies are providing LPN (Low Power Network) solutions. LoRaWAN is provided with a good network coverage in Switzerland.

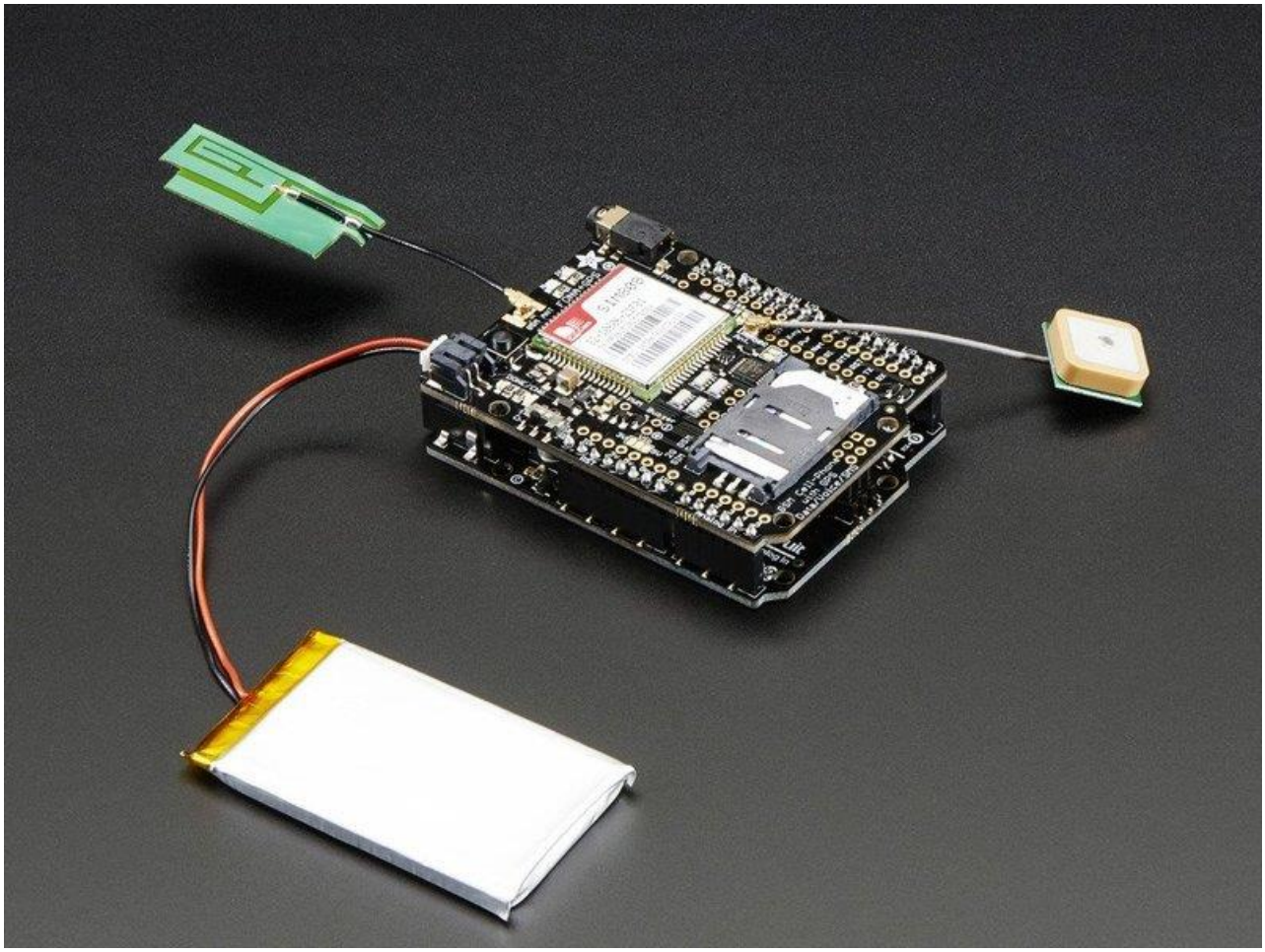


On the above picture you can see that the network coverage is very good within Switzerland, but as we can see, this is currently limited to Switzerland. There are in a lot of countries initiatives to build up a LoRaWAN coverage, driven by the LoRa Alliance.

For our use case such an approach would be promising in the future, right now there is the limitation of availability limited to Switzerland, which for our use case is not good enough. As a side note apart from that, the pricing of the LoRaWAN offering is not very lucrative.

I was further investing, looking for a way to communicate over the regular mobile telecommunication networks, which would have a very good international coverage.

Looking for devices with mobile communication network access and GPS, I came across this device: *Adafruit FONA 808 Cellular + GPS Shield for Arduino*. This provides GSM quad-band to send and receive GPRS data (TCP/IP, HTTP, etc.) and a fully integrated GPS.



Considering that there are plans to shut down GSM networks in some regions, a 3G chip may be considered as well, although even for 3G networks there are discussions for shutting them down. However, for the time being, it seems GSM is up and running in Europe. I decided to order this device for further investigation.

Assembly

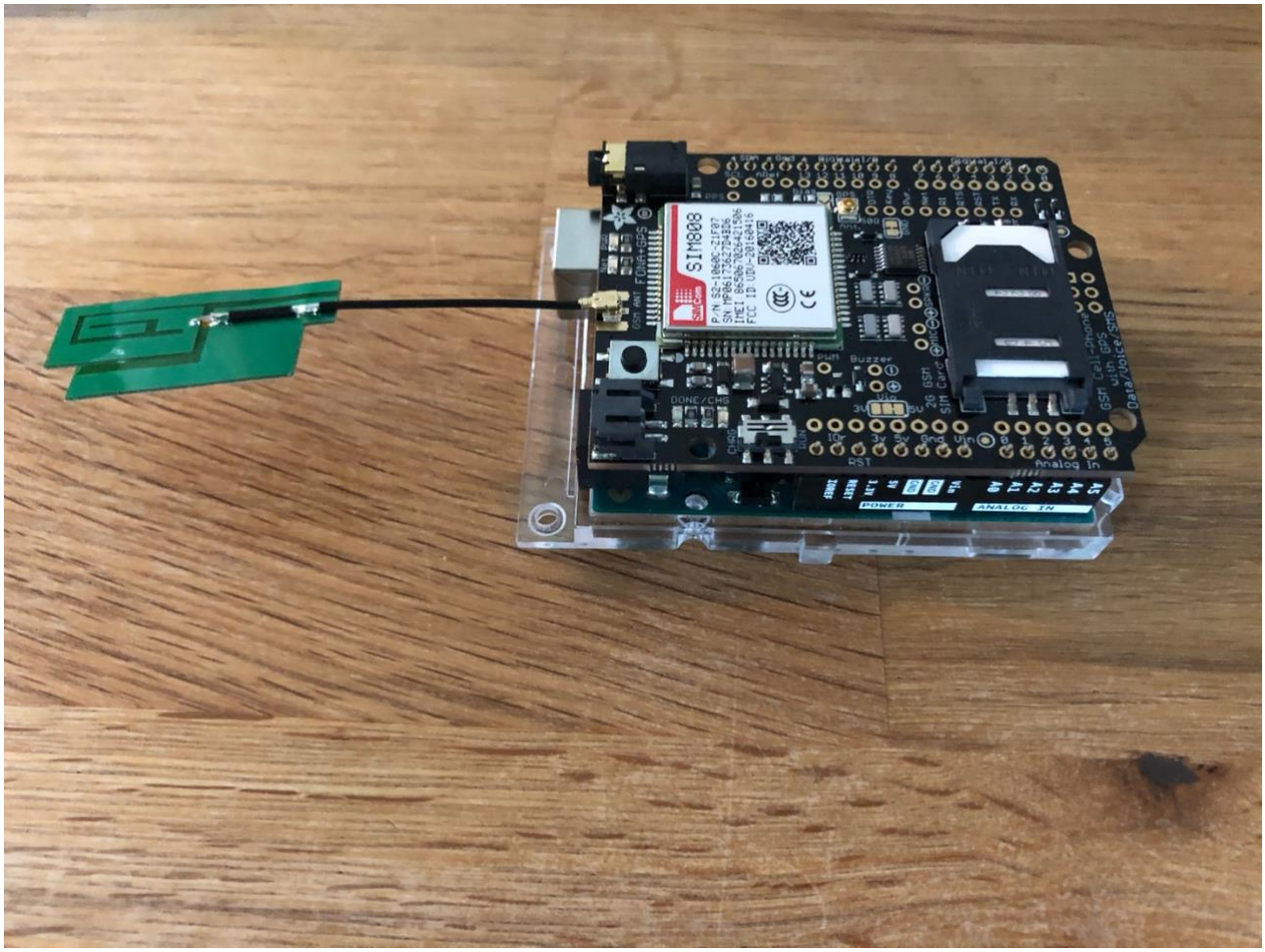
For the assembly you need a couple of additional parts, a battery, antennas and a SIM card.

Online documentation for the device can be found here: https://learn.adafruit.com/adafruit-fona-808-cellular-plus-gps-shield-for-arduino/assembly

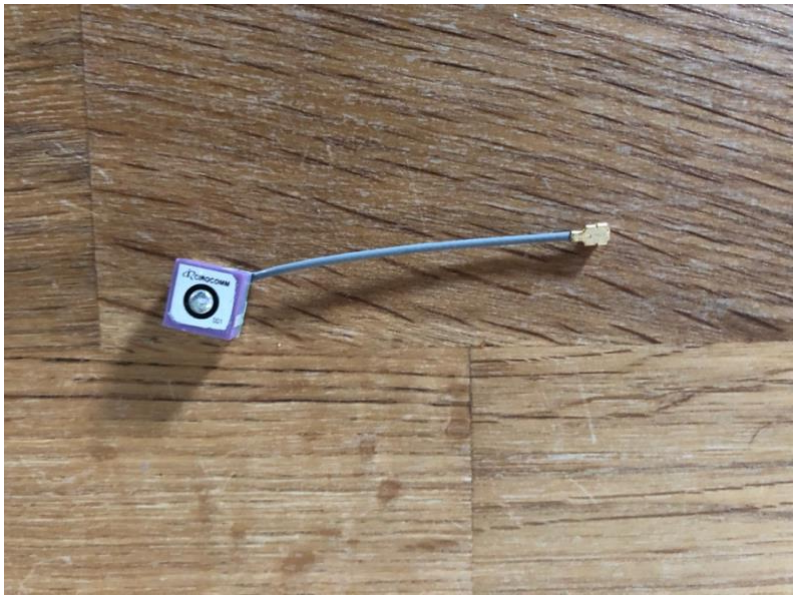
Implementing the device

The FONA module is a shield for Arduino, I was implementing it with an Arduino One.

You need to mount the FONA onto the Arduino and you need to put an antenna on it for mobile network access. You also need to install a SIM card, which is 2G capable. A regular 3G SIM card from Coop Mobile did work for me (which is easy to say afterwards, but beforehand it was quite unclear).

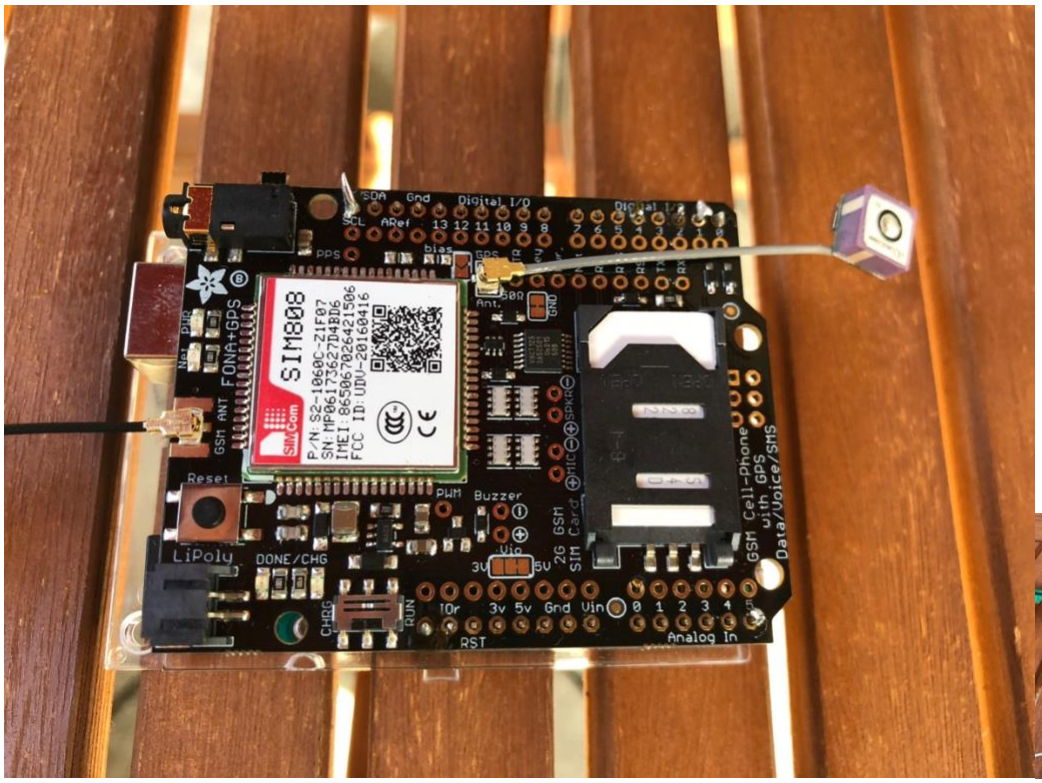


You then need to attach another antenna for the GPS, a passive GPS antenna.

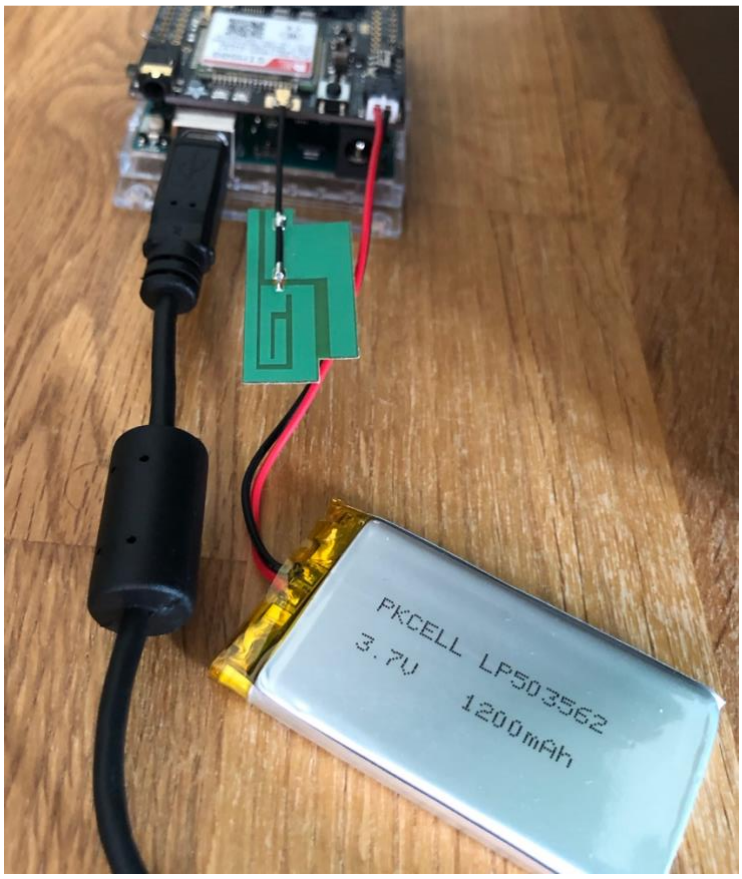


After that is all done, the problem is, you need to solder a lot of pins between the Arduino and the shield, which is a challenge for someone with no experience in soldering!

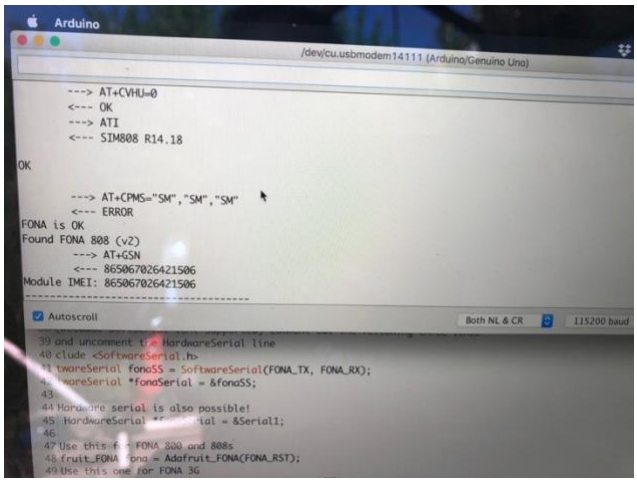
The first pins were a challenge, as they are pretty close to each other. Well, actually all pins!



You need to be patient and solder each pin, one by one with care. Also, you need to connect a lipoly battery to get it to work. I have chosen to use a 1200mAh battery.

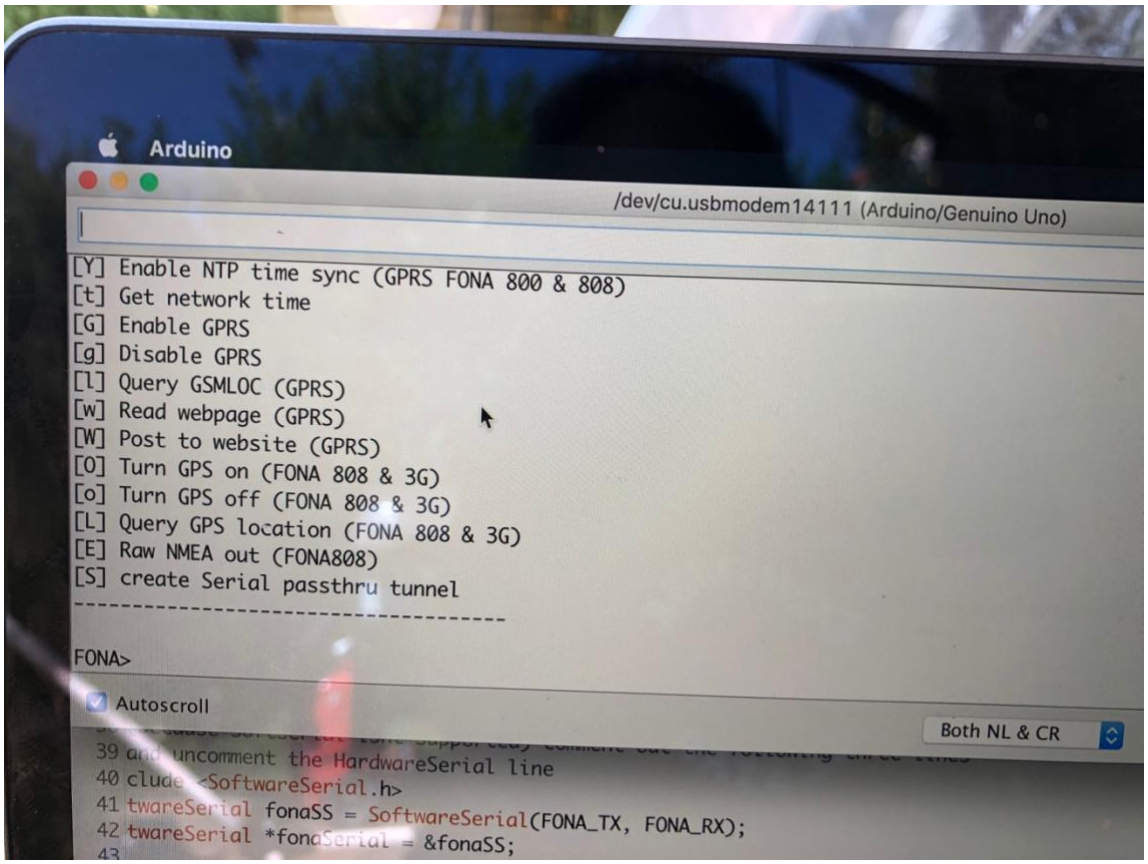


You know that you are done by the time you are able to connect via the Arduino IDE and if you are able to connect to the device by the FONA library provided by Adafruit.

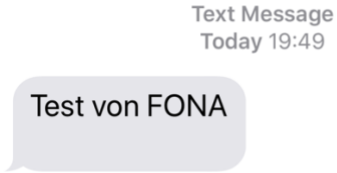


And hopefully you have some steady glooming and blinking LED’s!

Once that is done, you can start experimenting with the device. The provided FONA library with the test program is a great starting point. You can test every functionality of the board. Getting network access is a bit of a hurdle, as for some reason the SIM unlocking is not quite stable. Sometimes you are able to connect to the network and provide a PIN, but sometimes it seemed like the network locked me out. You need to be patient and after a couple of hours, you are all of a sudden able to get connected again.



You start with the simple things and getting through a SMS was the first success experience.



You also need to provide an APN name for GPRS access, which was not very obvious to find out for the provider I was using. And once you have been able to successfully enable GPRS, you need to enable the HTTPS redirect, since nowadays HTTPS is standard, otherwise you will never succeed to do a successful HTTP request from this device.

After overcoming all these hurdles, finally you are able to write a program! If you are, like I am, an experienced programmer, this won't be too hard. I was programming c++ for years, so it felt familiar, although you miss some of the convenient things, e. g. like STL or even boost.

As this is Arduino based, you find tons and tons of documentation, tutorials and other help on the internet. You most probably find something very similar to what you are doing. This obviously also applies to the tracking use cases. One very nice piece of code I came across was this one:

https://github.com/botletics/SIM7000-LTE-Shield/blob/master/Code/examples/IoT_Example/IoT_Example.ino

Studying this code was very beneficial and eye-opening.

GPS Experiment Time-To-First-Fix

If you start up the module it takes some time to get a position. In my tests it took a couple of minutes.

It starts with providing only timestamps. It shows information like this:

```
       ---> AT+CGNSINF
       <--- +CGNSINF:
Reply in format:
mode,fixstatus,utctime(yyyymmddHHMMSS),latitude,longitude,altitude,speed,course,fixmode,
reserved1,HDOP,PDOP,VDOP,reserved2,view_satellites,used_satellites,reserved3,C/N0max,HPA
,VPA
1,0,20180707162239.000,,,,0.00,0.0,0,,,,,,3,0,,,27,,
```

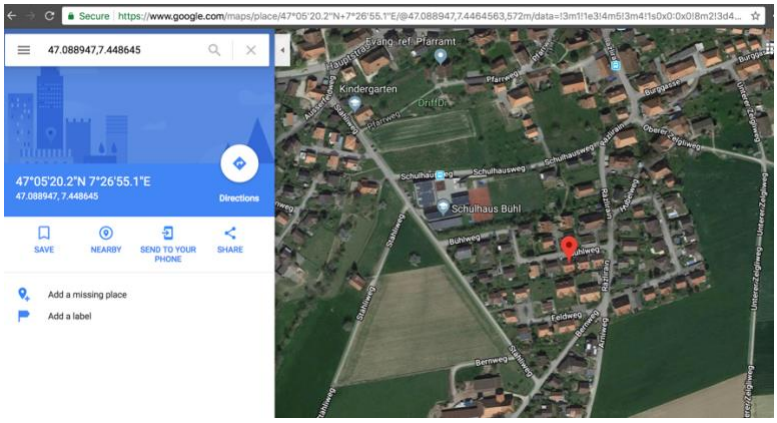
Once you get coordinates in lat/lng, it looks like this:

```
       ---> AT+CGNSINF
       <--- +CGNSINF:
Reply in format:
mode,fixstatus,utctime(yyyymmddHHMMSS),latitude,longitude,altitude,speed,course,fixmode,
reserved1,HDOP,PDOP,VDOP,reserved2,view_satellites,used_satellites,reserved3,C/N0max,HPA
,VPA
1,1,20180707163259.000,47.088690,7.448735,490.800,8.59,155.8,1,,1.9,2.1,1.0,,12,5,,,32,,
```

After some minutes, the position gets more and more accurate (you can observe the used satellites count to go up, from 0, to 5 and up to 7):

```
       ---> AT+CGNSINF
       <--- +CGNSINF:
Reply in format:
mode,fixstatus,utctime(yyyymmddHHMMSS),latitude,longitude,altitude,speed,course,fixmode,
reserved1,HDOP,PDOP,VDOP,reserved2,view_satellites,used_satellites,reserved3,C/N0max,HPA
,VPA
1,1,20180707164449.000,47.088947,7.448645,515.500,1.78,148.8,1,,1.5,1.7,0.9,,11,7,,,32,,
```

Which was pretty much exactly my position, as you can validate in a tool, e. g. Google Maps:



Device GSM

Remember, for getting a network registration you need to install a 2G SIM card, for which the prepaid SIM cards in Switzerland are still capable of (or I was just lucky enough to have chosen a card by chance that was capable).

```
       ---> AT+CREG?
       <--- +CREG: 0,0
Network status 0: Not registered
FONA> U
Enter 4-digit PIN
XXXX
Unlocking SIM card:        ---> AT+CPIN=XXXX
       <--- OK
OK!
FONA>
+CPIN: READY

Call Ready

SMS Ready

n
       ---> AT+CREG?
       <--- +CREG: 0,1
Network status 1: Registered (home)
```

As mentioned, the FONA also supports GPRS, which opens the door to TCP/IP and thus HTTP, as well as MQTT. It is possible to do HTTP REST calls, which will allow us to send the position of the device to the cloud. For calls to HTTPS it is necessary to configure the FONA API accordingly, so that it follows the redirects over SSL, which is possible over the *setHTTPSRedirect* setting. We can achieve the same thing over MQTT publish and subscribe, by using an already existing library, which can be used with the FONA: https://github.com/adafruit/Adafruit_MQTT_Library.

Indoor positioning

Indoor the GPS cannot get a position (which can easily be experienced with a FONA). In this case we have to fall back to the GSM location, which we are able to retrieve as well. This happens over the CIPGSMLOC, which is exposed over the FONA API *getGSMLoc*.

```
       ---> AT+CIPGSMLOC=1,1

       <--- +CIPGSMLOC:

7.470947,47.084225,2018/07/09,23:07:57
```

Device power supply

The tracker uses quite some power, as we have GSM and GPS on it. FONA needs a lipoly battery to work, it is supposed to work with a 500 mAh, but I have chosen the to use a 1200 mAh. The battery can be charged via the Arduino. When loaded, the circuit can be run over the battery by flipping a switch. But still, even with a big lipoly battery the device doesn't keep up for too long, a couple of days max. As elaborated previously, we need a much longer uptime for the device. Thus, we need a power supply, that keeps up the device a long time. That's why I was looking for solar panels, which could be used to power the device. I found a very nice solar researchable battery, which delivers the 5V necessary to load and power the Arduino/FONA. This is a commodity device, which can also be used for powering smartphones and other devices, while being on tracking or elsewhere without power supply for a longer period of time.

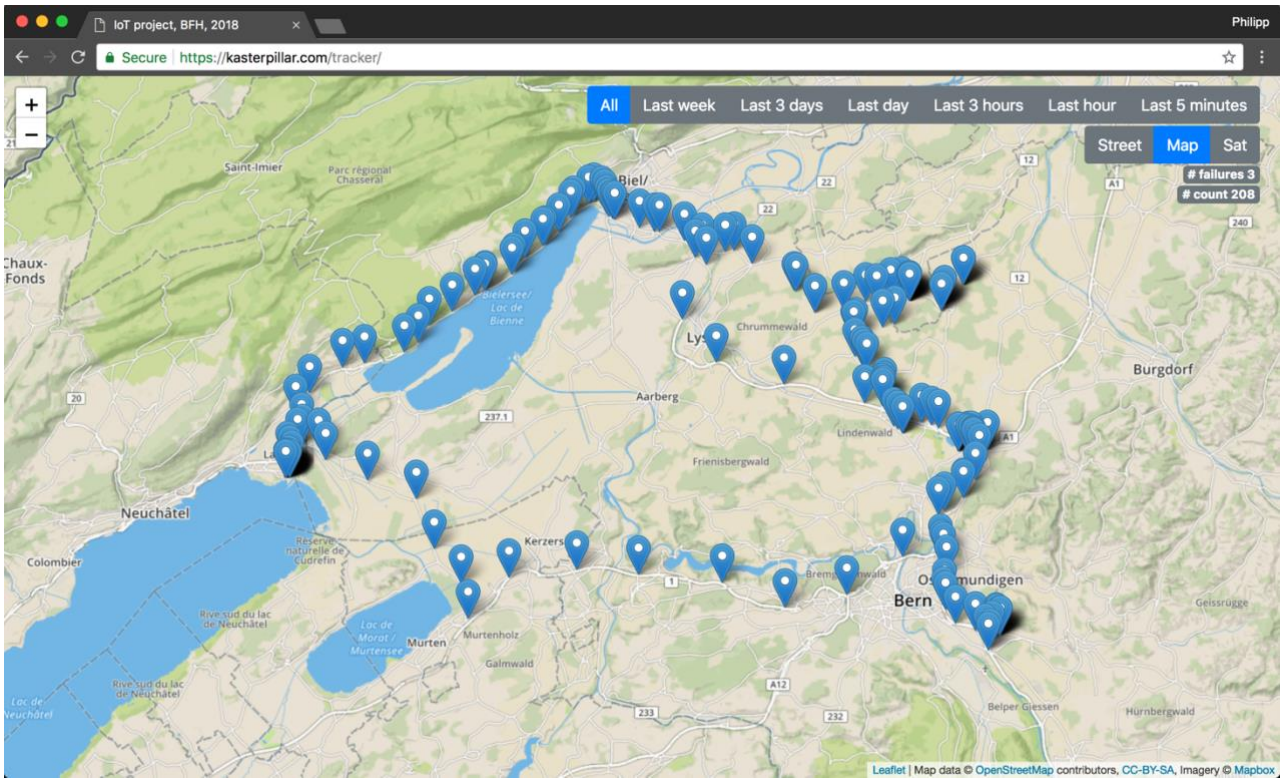
This device is working pretty nice to power this tracker. With enough sunlight it should be possible to power the tracker for very long periods of time. I will run a long-time test, but due to time constraints for writing this work, I cannot elaborate on that in detail yet.

Frontend/App

Now we are able to retrieve the position from the Arduino/FONA, the next step is: we need to implement the cloud side. Of course, a MQTT/HTTP based IoT cloud service such as from Azure, AWS or SIOT could be used. I decided to do this with my own web server in a first step. The server component provides a simple REST API, which receives the following values:

- Tracker id
- Lat/Long
- Timestamp
- Measurement parameter (GSM or GPS)

The values are stored in a database. The client can fetch these values as well over a REST API to the web app, which displays them on a dashboard. I implement a minimalistic web page, which displays the coordinates sent to the web service from the Arduino/FONA device.



Of course a wealth of other possibilities for integration exists, for which I had not enough time to look into: https://adafruit.io, https://dweet.io and https://freeboard.io or the https://thingsboard.io – of course besides all the other big players in this field.

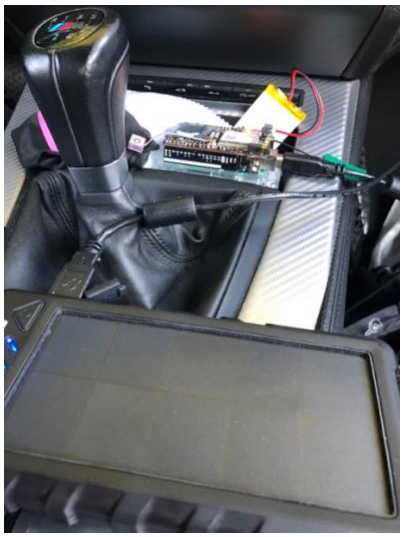
The essential parts of the code I was using can be found here: https://github.com/msekast/Tracker.

Conclusion

The device I was experimenting with, was very well working and convenient to program. Creating a tracker is possible and it runs quite stable. Still there are a couple of problems I see, w. r. t. implementing such a tracking system as described in the introduction of this paper:

- High costs for devices
- Still I cannot tell how long such a solar powered tracking device will keep up (but I will test it going forward)
- GSM dependencies, danger of network shutdown
- Device is too big, but it could be made much smaller, probably smaller than a smart phone

After this project work, I will definitely continue my experiments in this area, as it really looks promising.



Future work

Due to time constraints I haven't been able to dive into the topic of analyzing the data with machine learning approaches in the course of this work. The first implementation idea would be to connect the tracked object with an object in a BIM model, for which we know the following facts:

- Date at which the object has to be on construction site
- Planed location on the construction site

From that information we could calculate probable delays and misplacements and send alert notifications and display the state on a dashboard.