Electronic lock for European Cylinders

# Goals and Motivation

The goal of this project was to build an electronically operating lock cylinder for European locks. The lock should be operable by a smartphone and have a mechanical backup in case of a power failure (or a drain smartphone battery). Additionally, it is possible to restrict access for users to certain timeframes and all access attempts, whether successful or not, are logged.

To minimize the amount of required work to upgrade existing locks with our system, we chose to integrate all needed parts in or on the lock cylinder. The lock cylinder can be replaced in less than 5 minutes on regular doors, making the upgrade process really fast.

For the US market, there are already plenty of commercial and DIY solutions available for this tasks, but since European locks operate fundamentally different these solutions cannot be easily transferred to our project. Instead of setting up on those projects, we therefore chose to start from scratch. Additionally, our lock appears and operates as a normal lock cylinder from the outside. It therefore offers a compelling solution for upgrading doors with the possibility of electronic unlocking, without losing the security guarantees of a mechanical lock.

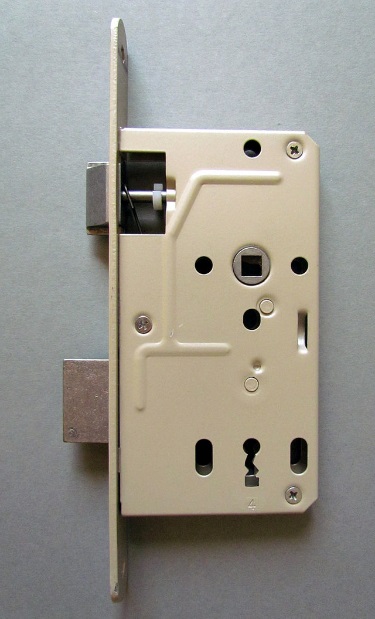


Illustration 1: European mortise lock (left) and American Deadbolt (right)

# System Overview

This project consists of two major parts: The app for the smartphone and a regular lock cylinder modified to be operable electronically. The modification of the lock cylinder must retain the mechanical operation since the user could be otherwise locked in in case of a power failure.

Since normal lock cylinders block if the key is inserted on the inside, we used a knob cylinder which by design does not have this problem. Additionally, by connecting to the shaft we are able to operate the cylinder electronically and mechanically.

The communication protocol between smartphone and lock cylinder is designed to be transport independent and while we chose Bluetooth Smart (formerly known as Bluetooth Low Energy) as the transport mechanism, any other transport mechanism could be chosen as well. The protocol is deliberately kept simple too keep the implementation requirements low on both sides.

# Hardware

The main part of the hardware is the lock cylinder. It needs to be modified to support the attachment of the motor while retaining full mechanical unlock capacities from both sides. This is necessary to avoid lock out or lock in in case of a power failure but also provides convenient operation for traditional users skeptical of any kind of electronic locking system.

To keep our hardware setup portable, we chose to integrate all required parts on the lock cylinder. This allows for fast modification and retains the impression of a normal lock from the outside. The lock motor is driven by an Arduino-compatible board which already has the required Bluetooth module on board.

## Lock cylinder

Most lock cylinders do not support turning with two inserted keys i.e. if a key is stuck from the inside, the lock cylinder will not support turning from the outside. While this security feature is wanted by many users, it is completely counterproductive for our project since we need to be able turn the lock cylinder from the outside even if the motor is attached on the inside.

One notable exception of this operating conditions are knob cylinders. Due to their nature, they always support turning from both sides since the knob is attached permanently. We therefore choose to use a lock cylinder with detachable knob for our project, hoping that we can add the gear interface for our motor and attach the normal knob on top. This way, we could achieve our goal of simultaneous mechanical and electronic operation.

As is it turned out, things weren’t that easy. The manufactures of the lock cylinder we got clearly didn’t plan for their product to be modifiable. While possible, the cylinder knob came of very reluctantly and a small blocking bar had to be removed. The required procedure will depending on the lock manufacture, but a (hopefully) useful hint is to search for a worm screw or a small bar holding the cylinder in place. More expensive versions sometimes use a shaft key, in this case a hole in the gear for mounting has to be planned for.

Once this was done we needed to find a way to attach our gear interface to the motor, since the shaft of the motor was completely round. We ended up reusing the hole used for attaching the knob, a process which is described in detail in the next section.



Figure : Lock Cylinder with mounted gear and knob

## Gear

In order to unite the lock cylinder with the motor, we needed a gear interface. The interface consists of one gear part for the motor and one for the lock cylinder. Thinking through the different combinations, we ended up trying a worm gear, mainly because it offered a high gear transmission ratio and a good mounting position for the motor.

Since the project institute had a 3D printer, we designed the gears using OpenSCAD on the computer and printed it out. After printing the first gear part it quickly turned out that the desired operation was not achievable with this gear combination because the newly printed worm gear needed too much force to turn. We therefore rethought our plans and resorted back to a classical bevel gear, which printed out fine.

To mount the gear on the cylinder lock, we reused a hole in the cylinder shaft which originally was used to fix the knob. We inserted a threaded rod in the hole, cut it to the right size and fixed it on both sides using nuts. Since cutting may damage the thread, we placed the nuts in their final position before cutting.

To minimize friction between the cylinder and the gear, we grinned the gears flat side.

The gear also includes 3 magnets which are used in combination with a hall switches to detect the lock cylinders current position and guarantee a safe states during locking and unlocking.

To attach the gear to the motor, a propeller adaptor was used. Originally coming from the model-making community, the adaptor allows to secure the gear on the motor shaft by using a screw.

At first, we tried to use a regular not-to-high speed model-making motor. However as it quickly turned out, the 5000 rpm provided by the motor were way too fast for the gear, so we needed to find a different solution. We ended up modifying a standard servo to be 360° turnable. Since most servos already include a high-ratio gearing, their rmps are mch lower.

This procedure involves opening the servo, taking its gearing apart and removing the blocking stab of one of its gears. It is described in detail on various howtos in the internet (just search for: “servo 360 degree”). Instead of driving the servo with the analog interface, we directly controlled the servo through its H-bridge. This allowed us to circumvent the problem of fixing the servo potentiometer to a fixed known value.

If a less “hacky” solution is desired, one can also use special servos called “sailwind servo”. Originally designed for operating the wind of model sailboats, these servos can turn more than 360° while still allowing position control. Since we drove the servo through the H-Bridge directly, the arduino firmware would have to be modified slightly to support these servos. However, that’s not too difficult, only the content of the *motorTurnRight*|*Left* and *motorStop* functions has to be changed.

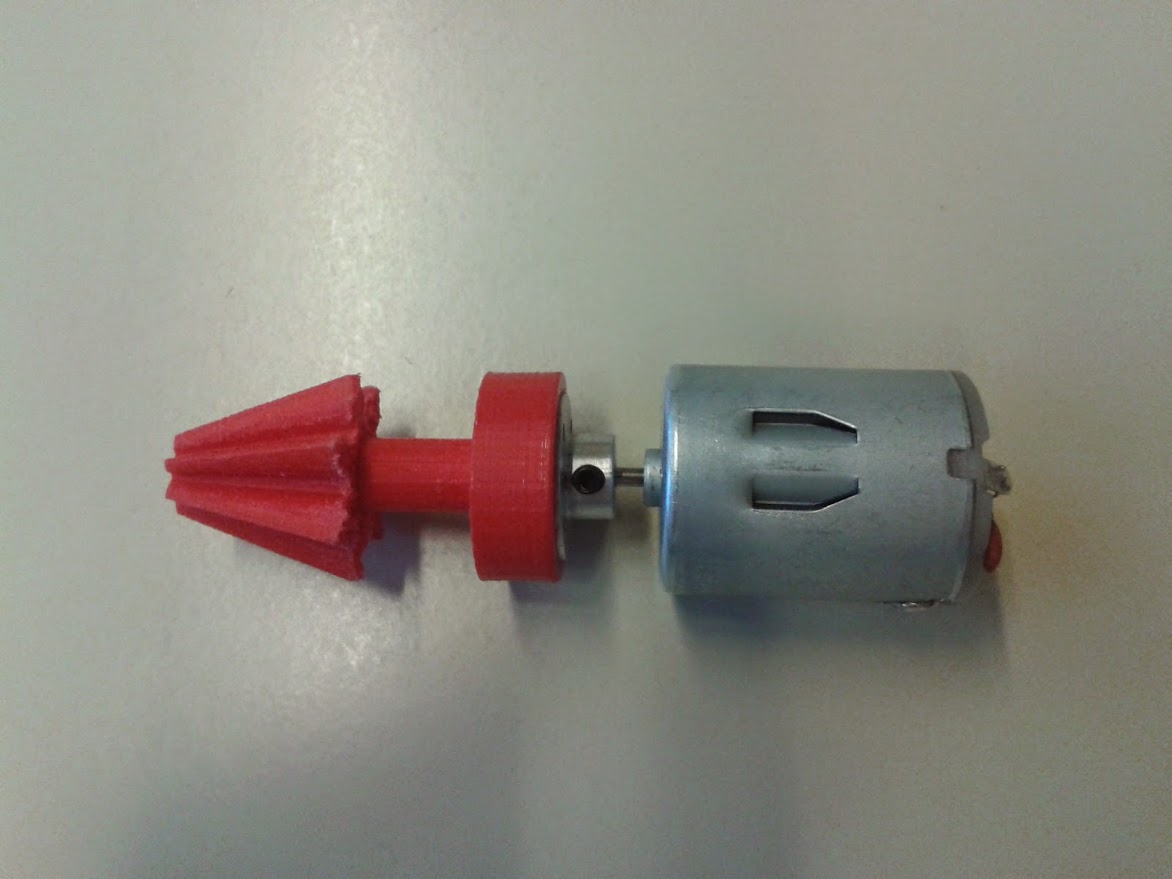


Figure : Initially used model-making motor with mounted gear



Figure 4: Open servo motor without gear

## Case

To fit all components to the door, we used a 3D printed case. Since our hardware is prototype, the case might be a little bit bulkier than it would have to be, however for our purpose it worked just fine. For demonstration purposes, we additionally built a case stander, so that the hardware could be operated without having to be mounted on a door.

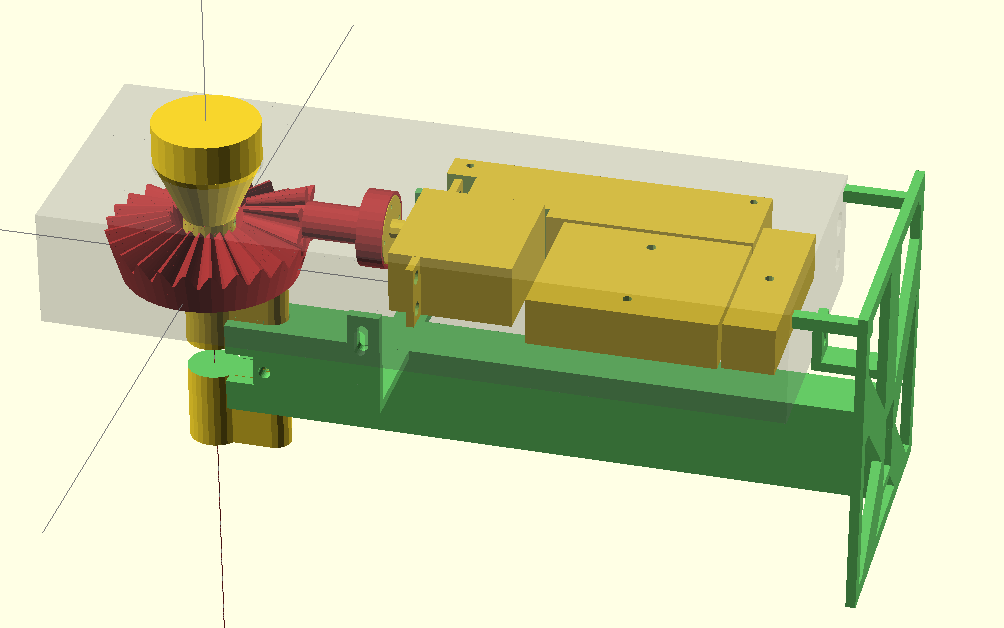


Figure : OpenSCAD 3D Model for the case with all components fitted

## Blendmicro

The Blendmicro is an Arduino compatible board with integrated Bluetooth Low Energy chip. It is responsible for the communication with the smartphone and operates the lock cylinder using an H-Bridge. To minimize the required space, an H-Bridge IC (L239D from Texas Instruments) is used. Additionally 3 Hall switches are required to monitor the position of the lock cylinder. They react to magnets mounted on the lock cylinders gear. And finally, the setup also consist of a button to put the software in temporary registration mode.

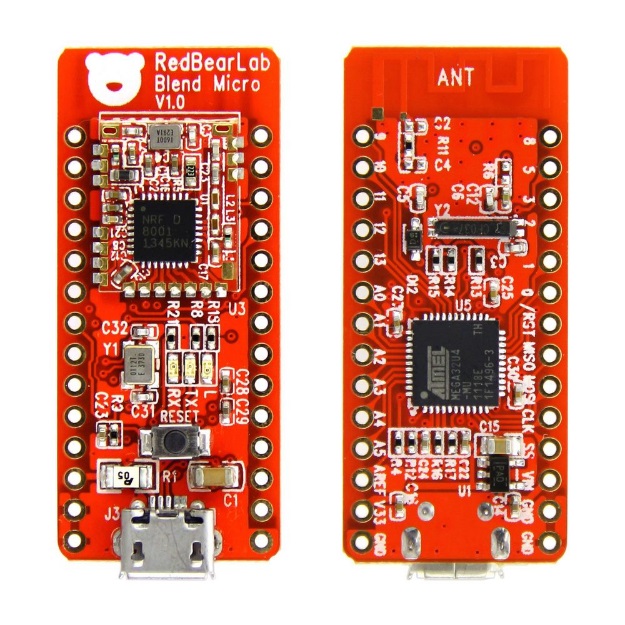


Figure : The Arduino compatible Blend Micro Board

# Software

The software comprises of the Android app and the Arduino operated cylinder. They are both connected through a simple communication protocol. The Android app does support multiple locks while the Arduino may only accept one connected smartphone at the time. Multiple smartphones can be registered of course.

Instead of using a standard Arduino with additional Bluetooth smart shield, we opted to use the Blendmicro, an Arduino compatible board which already has the required Bluetooth chip onboard. While the producers of Blendmicro offer a low level Bluetooth library, we used an open source third party library to easily support higher level Bluetooth constructs.

## Arduino

The main purpose of the Arduino is to drive the motor operating the lock cylinder and respond to Bluetooth requests. Additionally, it tracks the current position of the lock using the gear magnets in combination with hall switches. User logging is currently not implemented.

The software sets up all required connections in the setup function and then continuously polls for Bluetooth messages in the loop function. Incoming messages are then handled by the configured handler functions. After a successful unlock request, the motor is driven using an H-Bridge chip.

## Android

The Android app is responsible for sending unlock requests to the lock and offering the user an offering easy to use UI. The UI consists of two screens: the main screen displaying the current lock status and offering unlock (MainActivity) and a register screen showing all lock cylinders in range (AddLockActivity). The app also makes use of a background service (BLService) for monitoring the Bluetooth connection and managing all communication with the lock cylinder.

Note that the smartphone running this app is required to support not only normal Bluetooth but Bluetooth Low Energy. At the time of this writing, this mainly includes new Android smartphones.

## Communication protocol

The protocol is built to ensure the smartphones identity while keeping the communication overhead to a minimum. A simple challenge-response scheme is used.

Table : Byte allocation for client request with random number

|  |  |  |  |
| --- | --- | --- | --- |
| Byte 1 | Byte 2 – 5 | Byte 5- 6 | Byte 7 - 20 |
| Command | Random | ID | HMAC |

The smartphone sends an authentication request to the client, which the client answers with a random number as a challenge. The client uses this challenge to draft a message consisting of desired command, random number, user id (which has been established during initial enrollment) and HMAC. As HMAC hash function, sha256 is used which is truncated to 14 bytes to minimize the message length.

After receiving the command, the lock verifies the user identity using the HMAC and if successful executes the command. A sequence diagram for this communication is shown in Figure 7.

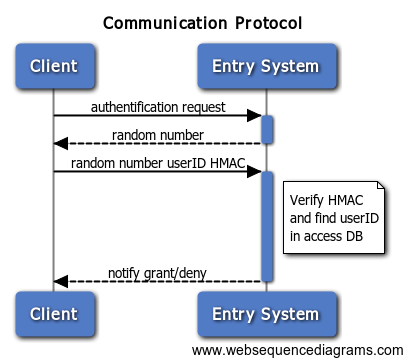


Figure 7: Communication between Smartphone (left) and lock cylinder (right) for unlocking

For the initial registration, the client sends a registration request to the lock. If the lock is ready to accept registration (this currently requires the user to press a button) it answers with a new user ID and key otherwise an error is signaled. This process is shown in Figure 8.

Note that if the key exchange is observed by an attacker, she could use that information to compromise security.

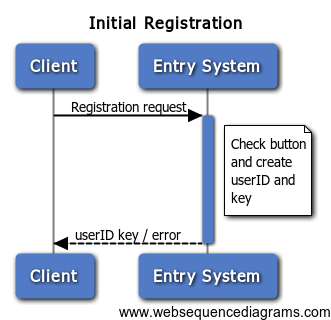


Figure : Registration protocol

# Summary and Outlook

In this project, we built an electronic lock for European cylinders operable by an android app using a Bluetooth low energy capable smartphone. We devised a protocol for secure communication, programmed the android app and arduino firmware and built the necessary hardware.

In the retrospective, we could have done a few things differently from the start. Instead of trying with a mode-making motor, we could have used a servo and a bevel gear instead of the “optimized” worm gear. We also had huge problems with our hall switch based positioning system which was not triggering all switches at the same time. A potentiometer based positioning would have been much easier.

This documentation exists to help people going down the same path to avoid the problems and pitfalls we encountered.

One of the main advantages of this project is that it is built completely modular. For example, access times can be implemented by reusing the unlock protocol and adding a “set access time” UI to the android app.

Possible extensions include remote unlocking and access logging from the internet. Integration with platforms like ifttt would be possible by using a low cost wlan chip or even celluar radio using newly emerging projects like the spark electron. The possibilities are endless, only the power consumption must be kept in mind.

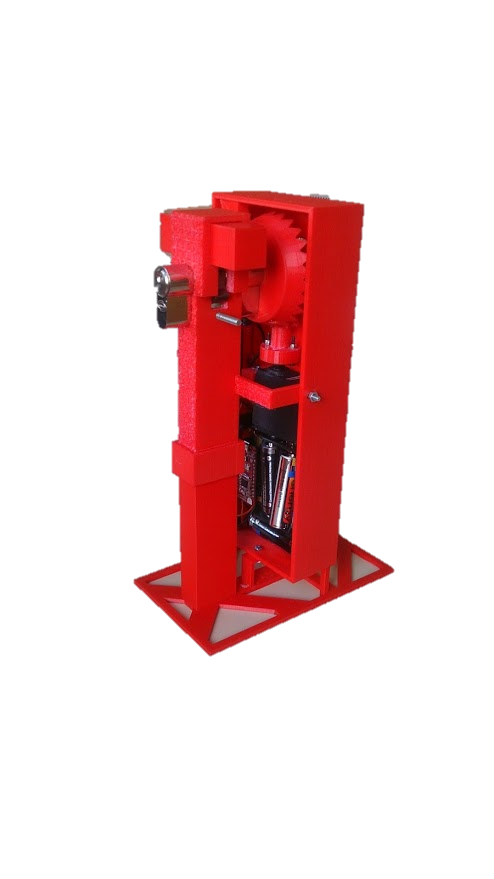


Figure : FInished lock prototype internals (left) and frontal (right)