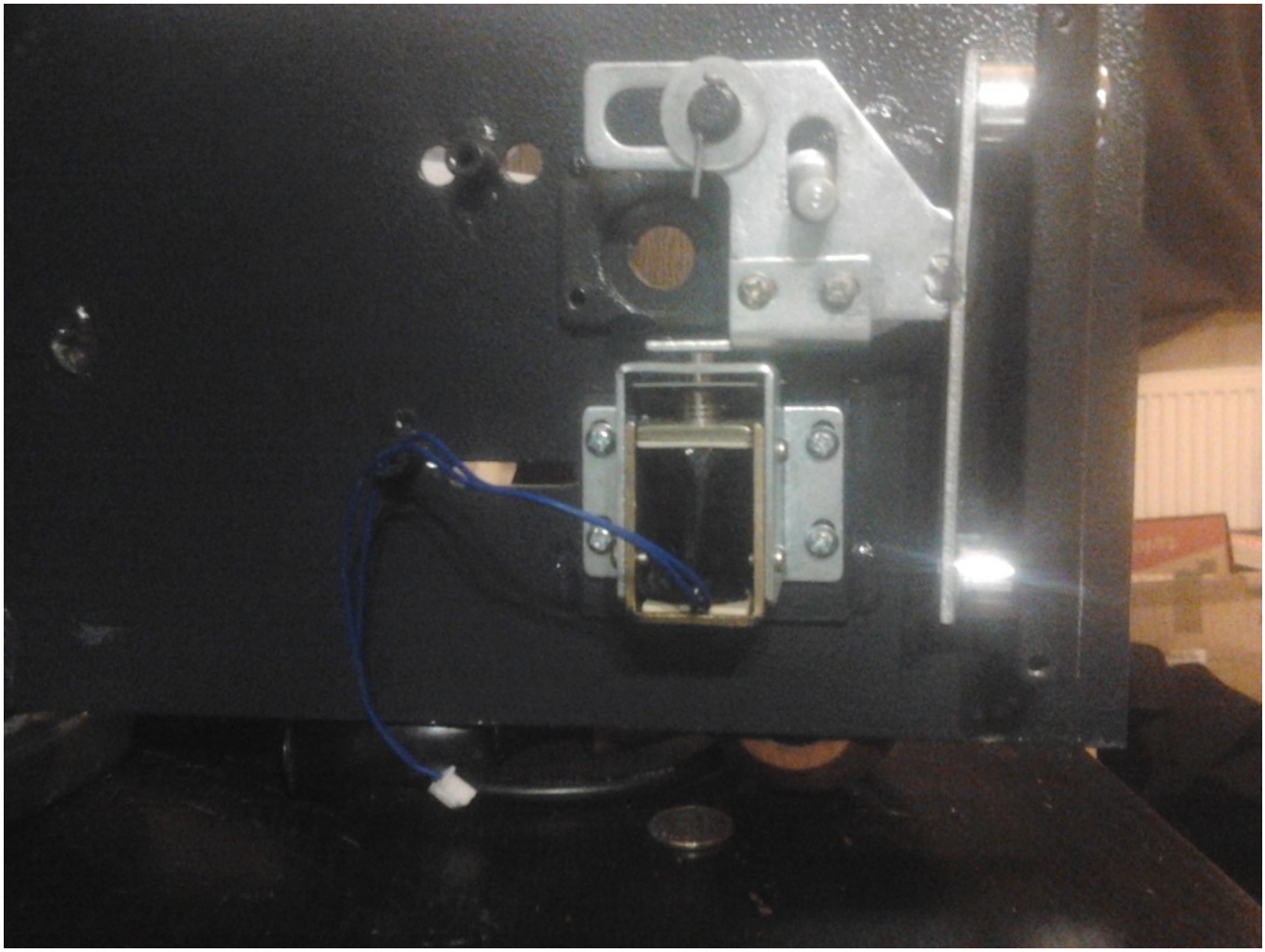
Witam! Jakiś czas temu od jednego z ircowych znajomych udało mi się wyrwać stary sejf. Wszystko było by pięknie gdyby nie fakt, że brak mu było elektroniki sterującym mechanizmem blokującym zamek. Pomyślicie, po co mi taki sejf?

Sam po otwarciu go zastanawiałem się co z nim zrobić. Tak po otwarciu. Dostałem go zamkniętym i nie wiedziałem jaki mechanizm tam się znajduje.



Przez wiele czasu, bo ponad pół roku służył jako szafka i podstawka do monitora. Kilka dni temu patrząc na niego z politowaniem wymyśliłem dla niego zastosowanie ambitniejsze niż tylko szafka. Mianowicie postanowiłem go przywrócić do stanu pierwotnego. No, może nie pierwotnego, ale zbliżonego do pierwotnego.

Po przegooglowaniu internetów okazało się, iż to co chce zrobić jest banalnie proste. Jak już pisałem brakowało jedynie elektroniki. Mechanizm blokujący był gotowy i sprawny. mechanizmem tym okazała się cewka indukcyjna zasilana prądem stałym 5v i potrzebująca cholernie mało amperów. Niewiele myśląc sięgnąłem po dobrze nam znana atmegę i zabrałem się do projektu. A oto co bedzie nam potrzebne.

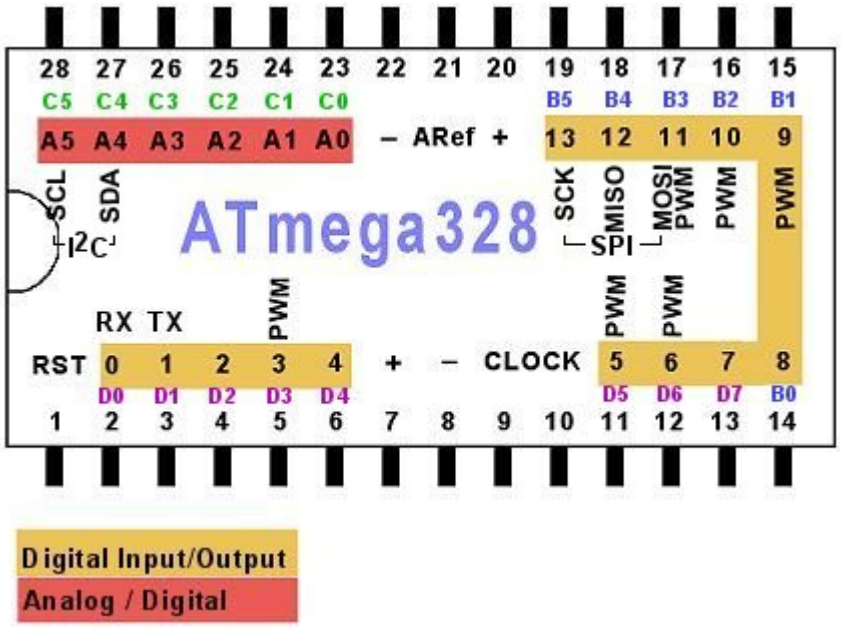
1. Atmega 328P.
2. Kryształ 16Mhz.

3. Zielona dioda led.
4. Czerwona dioda led.
5. Mosfet z kanałem N moze byc irf510
6. Czytnik rfid + conajmniej dwie karty/tokeny.
7. Coś czym będziemy zamykać/blokować – u mnie będzie to cewka indukcyjna z bolcem blokującym zamek.
8. Coś co chcemy zamknąć, szafka, sejf – u mnie będzie to sejf.
9. Moduł zasilania
10. Specjalnie zaprojektowana płytka pcb lub płytka do szybkiego montażu.

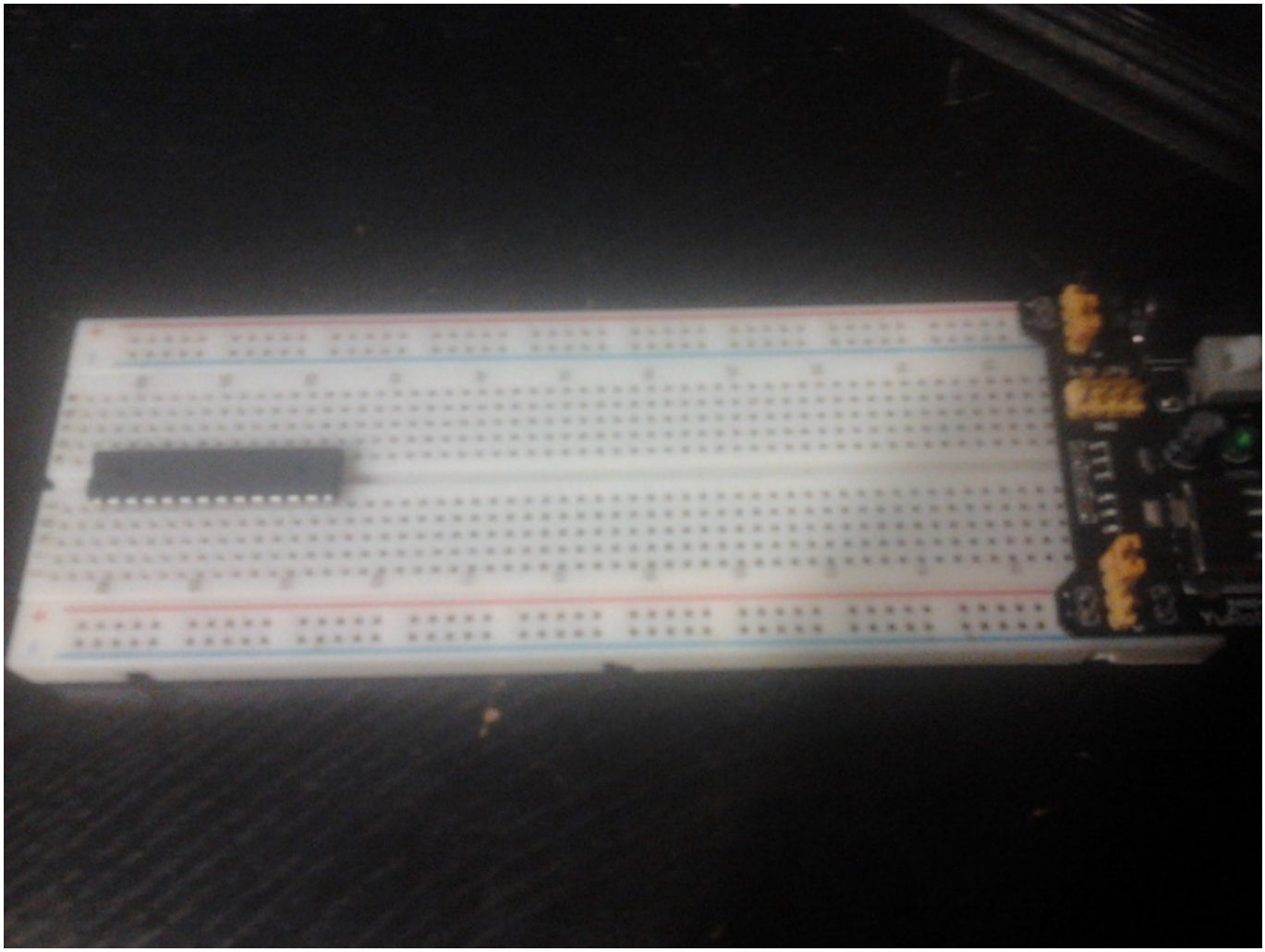
Oczywiście zamiast atmegi możemy użyć gotowego już arduino pro mini. Tylko po co wydawać pieniądze na coś co wlutujemy w układ?

Najpierw warto zacząć od tego jakich kart możemy użyć. Najszybciej uda się nam kupić czytnik kart pracujący na częstotliwości 13,56MHz. Podkreślić należy, że tym czytnikiem **NIE ODCZYTAMY** kart ztm, czy dostępowych np do drzwi w pracy.

Czas zacząć się planowaniem naszego zamka. Zgodnie z wszelakimi datasheetami łączymy elementy. Opiszę teraz na szybko co do czego. Ponownie odwołujemy się do pinoutu naszej atmegi.



Zaczniemy od umiejscowienia atmegi oraz modułu zasilania na płytce stykowej.



Następne kroki to:

1. Podłączenie diody zielonej do pinu A3 (sygnalizacja akceptacji karty).
2. Podłączenie diody czerwonej do pinu A2 (sygnalizacja karty niezgodnej z wzorem).
3. Podłączenie cewki indukcyjnej do pinu A1. Chwilowo do prototypownia użyjemy diody led zielonej. potem diodę zastąpi mosfet n irf510.

Trochę wiki:

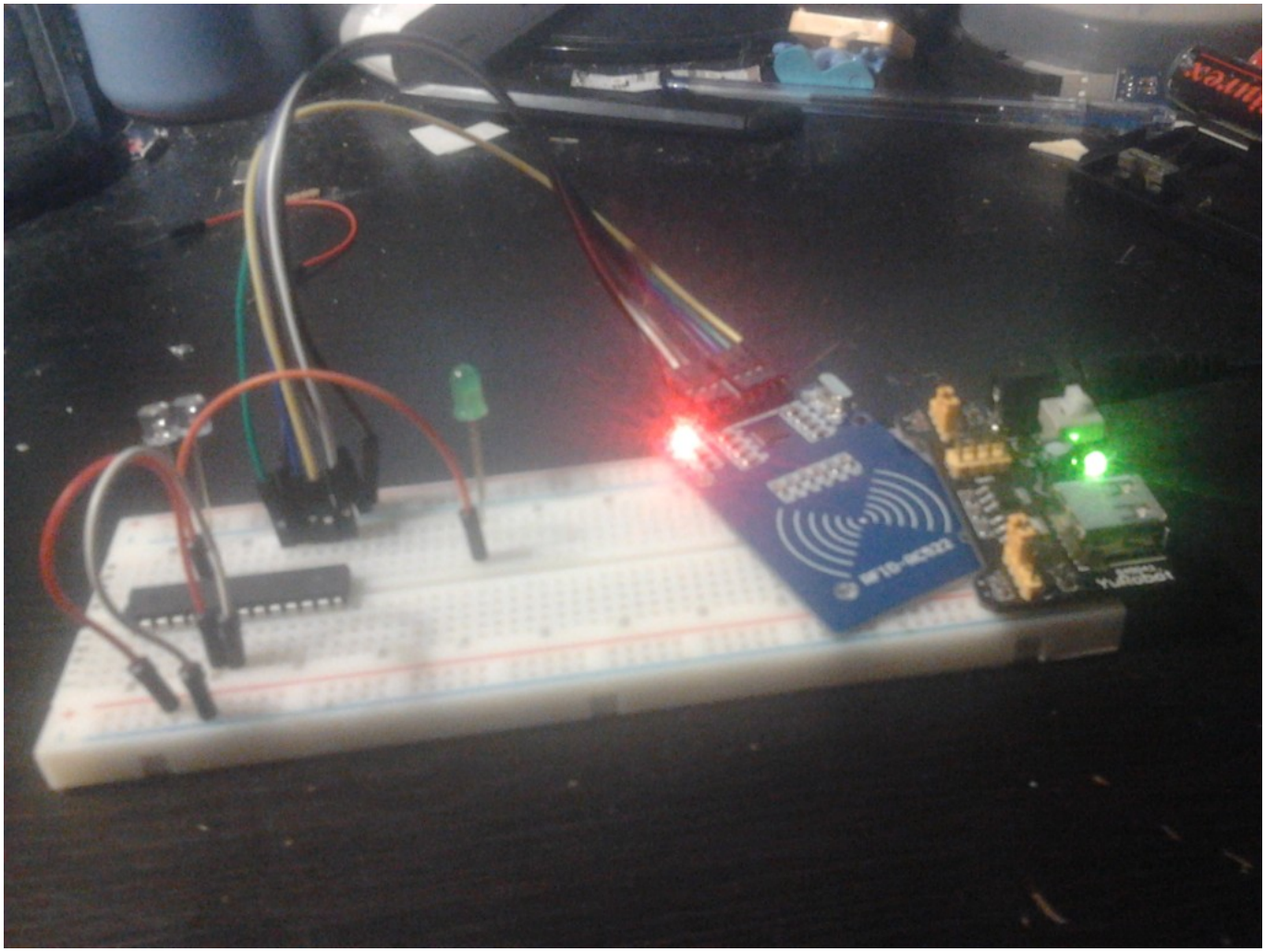
MOSFET (*Metal-Oxide Semiconductor Field-Effect Transistor*) – technologia produkcji tranzystorów polowych z izolowaną bramką i obwodów układów scalonych.

Jest to aktualnie podstawowa technologia produkcji większości układów scalonych stosowanych w komputerach i stanowi element technologii CMOS.
W technologii MOSFET tranzystory są produkowane w formie trzech warstw. Dolna warstwa to płytka wycięta z monokryształu krzemu lub krzemu domieszkowanego germanem.
Na płytkę tę napyla się bardzo cienką warstwę krzemionki lub innego tlenku metalu lub półmetalu,
która pełni funkcję izolatora. Warstwa ta musi być ciągła (bez dziur), ale jak najcieńsza.
Obecnie w najbardziej zaawansowanych technologicznie procesorach warstwa ta ma grubość pięciu cząsteczek tlenku.
Na warstwę tlenku napyla się z kolei bardzo cienką warstwę dobrze przewodzącego metalu (np. złota).
Układ trzech warstw tworzy prosty tranzystor lub pojedynczą bramkę logiczną układu procesora.

I znów krótko podsumujemy wiki. To co musicie wiedzieć. Kiedy podamy stan wysoki, czyli włączymy prąd na pinie mosfet do którego środkowej nóżki podepniemy będzie przewodził prąd.

4. Podłączamy nasz czytnik RFID

- SDA ->16
- SCK -> 19
- MOSI -> 17
- MISO -> 18
- RQ -> niepodłączony
- GND -> masa
- RST -> 15
- 3.3V -> Zasilanie 3.3V



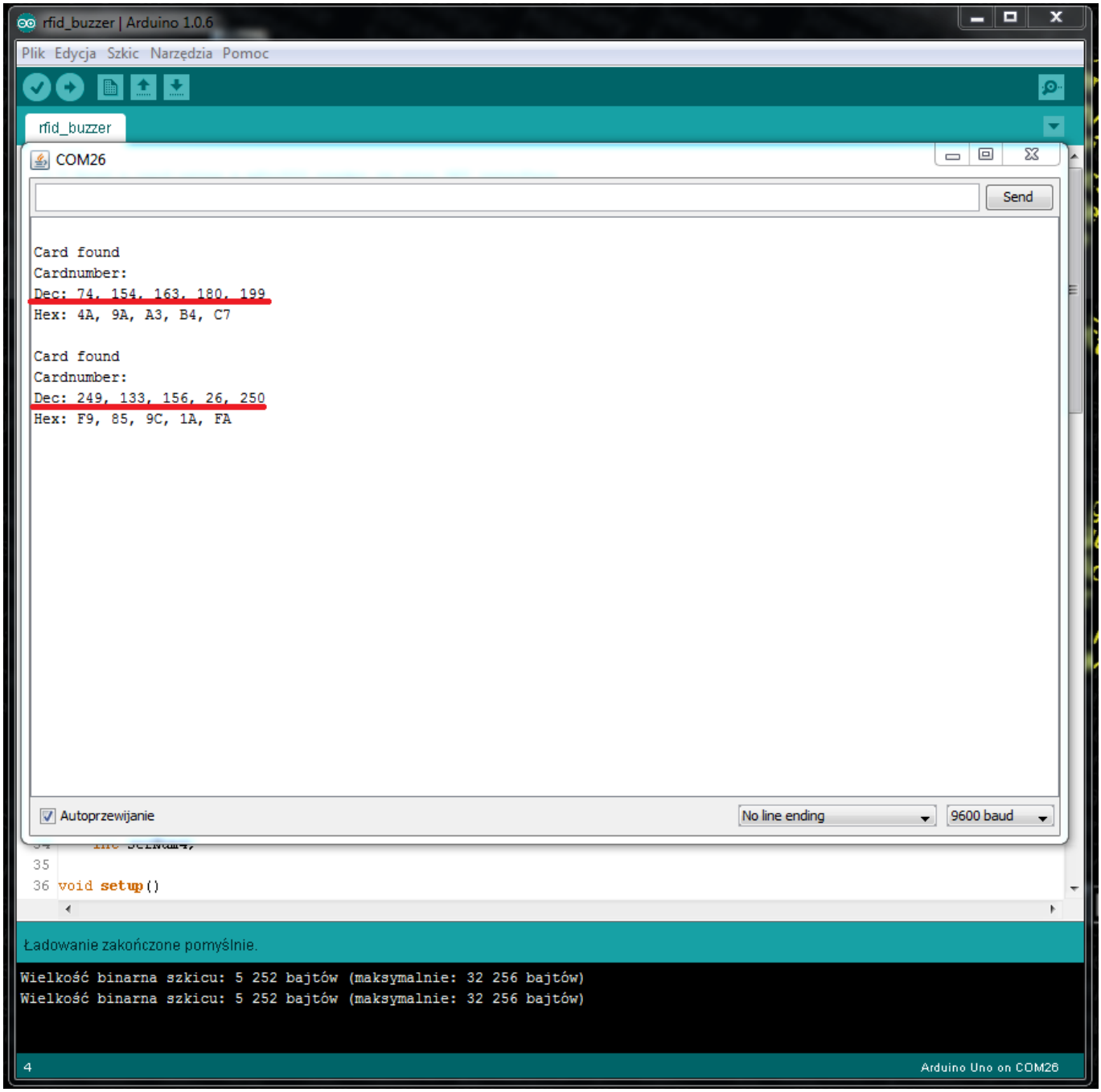
Całość powinna wyglądać jak na powyższym zdjęciu. Następnym krokiem będzie podłączenie programatora. Tutaj znów posłużę się arduino Uno jak w pierwszym arcie o arduino.

Docieramy do najciekawszego momentu. Mamy zbudowany czytnik + mechanizm zamka. Super. Ale jak go obsłużyć? Jakiego kodu użyć. Ja użyłem biblioteki RFID. W niej można znaleźć szkic rfid_buzer. Za jego pomocą możemy odczytać numer karty. Aby to zrobić należy grać nasz szkic do arduina połączonego bezpośrednio do czytnika rfid bez atmegi. Następnie w serial monitorze sprawdzić numer karty. Na szczęście dysponuje drugim arduinem które posłuży nam jako czytnik kart. Oczywiście warto znać numery swoich kart. Ułatwia to prace z nimi.

Połączenie rfid -> arduino Uno

- SDA ->10
- SCK -> 13
- MOSI -> 11
- MISO -> 12
- RQ -> niepodłaczony
- GND -> masa
- RST -> 9
- 3.3V -> Zasilanie 3.3V

Wgrywamy na arduino szkic rfid_buzzer z przykładu. Po wgraniu klikamy na Narzędzia – > Monitor portu szeregowego i odczytujemy numer karty.



Na powyższym screenie widać numery kart w zapisie decymalnym oraz hexadecymalnym. Nas interesuje numer pierwszej karty, jej użyjemy jako klucza do sejfu.

A więc czas zacząć programować! Co już wiemy?

Znamy numer naszej karty. Możemy go zsumować i w ten sposób ułatwić sobie prace, ale zarazem zmniejszymy wartość zabezpieczenia jakim jest nasz sejf. Dlaczego? To proste

ale już tłumacze.

W tym momencie warto odwołać się do artykułu niebezpiecznika, który już jakiś czas temu pisał o kolizjach identyfikatorów kluczy gpg, ale nie to jest teraz najważniejsze.

Załóżmy że zsumowaliśmy numer pierwszej karty (74+154+163+180+199) otrzymaliśmy wynik 770. Osoby lubiące sudoku i inne łamigłówki zapewne już zauważyły na czym polega problem. Otóż jeżeli weźmiemy karty z numerami:

74:254:63:180:199

154:74:163:189:190

199:189:163:74:154

Otrzymamy identyczną sumę, czyli 770. Oczywiście, możemy zrobić aby program sprawdzał sumę oraz ostatnią wartość, co zmniejszy ilość kolizji ale nadal będą one bardzo prawdopodobne. W tym momencie jedynym sensownym rozwiązaniem jest zastosowanie metody porównywania wzorca, czyli najprostszy na świecie if, który sprawdzi czy dana karta ma ten sam id co karta wpisana w programie. W tej wersji programu posłużymy się wpisaniem karty w kod. W wersji 2.0 pewnie dorzucę do funkcjonalności opcję, aby numer karty był przechowywany w pamięci EEEPROM, co pozwoli nam na to by nasza karta była w pamięci nawet po zmianie softu oraz aby ładniej to wyglądało.

Myślę, że tyle teorii na razie starczy. Czas przejść do praktyki. Pozwoliłem sobie przygotować wcześniej kod programu (zamek-rfid). Teraz przejdźmy do jego analizy.

Source code

```
#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

RFID rfid(SS_PIN, RST_PIN);

int serNum0;
```

```
int serNum1;
int serNum2;
int serNum3;
int serNum4;
int led1Pin = A3;
int led2Pin = A2;
int indPin = A1;
void setup()
{
  SPI.begin();
  rfid.init();
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
  pinMode(indPin, OUTPUT);
}
void loop()
{
  if (rfid.isCard()) {
    if (rfid.readCardSerial()) {
      if (rfid.serNum[0] == 74
        && rfid.serNum[1] == 154
        && rfid.serNum[2] == 163
        && rfid.serNum[3] == 180
        && rfid.serNum[4] == 199 )
      {
        digitalWrite(indPin, HIGH);
        digitalWrite(led1Pin, HIGH);
        delay(2000);
        digitalWrite(led1Pin, LOW);
        digitalWrite(indPin, LOW);
      }
      else
      {
        digitalWrite(led2Pin, HIGH);
        delay(100);
        digitalWrite(led2Pin, LOW);
        delay(100);
        digitalWrite(led2Pin, HIGH);
```

```
        delay(100);
        digitalWrite(led2Pin, LOW);
        delay(100);
        digitalWrite(led2Pin, HIGH);
        delay(100);
        digitalWrite(led2Pin, LOW);
      }
    }
  }
  rfid.halt();
}
```

Gdy mamy już napisany program i mamy prototyp, możemy zabrać się do zrobienia płytki. Oczywiście zanim się za to zabierzemy musimy skompletować potrzebne nam rzeczy.
W następnym artykule pokaże działanie prototypu zamka, oraz przygotuję produkt końcowy do zastosowań na skale domową.