Smart Door Lock

Student: Puflene Alex-Costin
Grupa: 332CC

Introducere

Proiectul consta intr-o incuietoare “inteligenta”, al carei scop este de a deschide usa de la casa printr-o parola generata pe telefonul mobil, scapand astfel de necesitatea cheii.

Descriere generală

Arduino-ul va controla deschiderea usii printr-un servomotor, care va invarti cheia in usa pentru inchidere / deschidere. Codul de acces OTP va fi generat printr-o aplicatie pe telefon, apoi transmis de telefon prin bluetooth, verificat mai apoi de placuta, care va primi pachetul cu ajutorul modulului ZS-040 (Bluetooth). Incuietoarea va avea un buton pe partea din interior pentru a facilita deschiderea usii fara cod la iesire.

Hardware Design

Lista de piese necesare:

- Arduino UNO ATMega328P;
- Modul Bluetooth ZS-040;
- Servomotor SG90;
- Modul RTC PCF8563;
- Buton;
- Rezistenta 10K;
- Cabluri.

Software Design

Biblioteci folosite:

- “Servo.h” - pentru controlarea servomotorului
- “Wire.h” - pentru comunicarea I2C, dependinta a “I2C_RTC.h”
- “I2C_RTC.h” - biblioteca third party, folosita pentru comunicarea cu modulul I2C
- “time.h” - folosita pentru structuri ce modeleaza timpul curent, dependinta a “TOTP.h”
- “sha1.h” - biblioteca third party, contine functii criptografice pentru a genera hash-uri SHA1; dependinta a “TOTP.h”
- “TOTP.h” - biblioteca third party, folosita pentru a genera coduri TOTP conform standardului RFC 6238
- “SoftwareSerial.h” - pentru definirea unei interfete seriale software, pentru comunicarea cu modulul Bluetooth
- “string.h” - pentru strcmp, pentru a compara codul generat pe placuta cu cel primit

Pentru inceput, am conectat butonul la placuta, folosind o rezistenta de pull-down de 10K (inainte sa realizez ca pot sa activez din cod rezistenta interna de pullup a placutei :)) si am testat ca placuta primeste semnal, aprinzand ledul intern. Apoi am conectat servomotorul, am testat ca functioneaza bine biblioteca, dupa care am incercat sa actionez servomotorul din buton cu un delay de 10s (actiunea de descuiat / incuiat usa), folosindu-ma de o intrerupere pentru a nu ocupa loop-ul principal, si am decis sa folosesc o intrerupere de tip PCINT (deoarece mi-a fost lene sa conectez butonul la intreruperile de tip INT0 sau INT1, intrucat deja facusem montajul electric, iar pe deasupra mi se parea mai interesant sa incerc sa folosesc intreruperi de tip PCINT intrucat presupunea lucrul cu registri). Am activat intreruperile pentru PORTB in registrul PCICR, fiind cel corespunzator pinului la care am conectat butonul, iar apoi am activat intreruperea specifica pinului respectiv prin modificarea registrului PCMSK0. In functia pentru intreruperea din PCINT0_vect (activata de PORTB), am aplicat si o metoda de debouncing prin a verifica daca exista o diferenta de minim 250ms intre activarea curenta a intreruperii si cea precedenta, si folosindu-ma de o variabila logica pentru a bloca intrarea pe intrerupere odata ce a fost activata, pana la expirarea timpului de 10 secunde.

Ulterior, am incercat sa folosesc un timer ca sa simulez descuiatul usii, asteptat 10 secunde, apoi incuiat; aici a fost punctul in care am aflat ca functille de delay/millis nu functioneaza cand se executa cod de intreruperi :). Ca sa rezolv aceasta problema, pe intrerupere am declansat un timer pe care am atasat o intrerupere modificand registrii TIMSK2, TCCR2B (setez prescaler la 256 prin setarea bitilor CS21 si CS22) si OCR2A (setez la 0xF9 pentru a numara pana la 4ms), iar apoi pe intreruperea asociata timer-ului numar 2500 de intrari pentru a ajunge la intervalul dorit de 10 secunde, moment in care dezactivez intreruperea pe timer. Initial aici folosisem timer-ul 0, pana sa realizez ca intreruperea pentru timer 0 este folosita de functii interne ale Arduino-ului, si mi se bloca complet placuta.

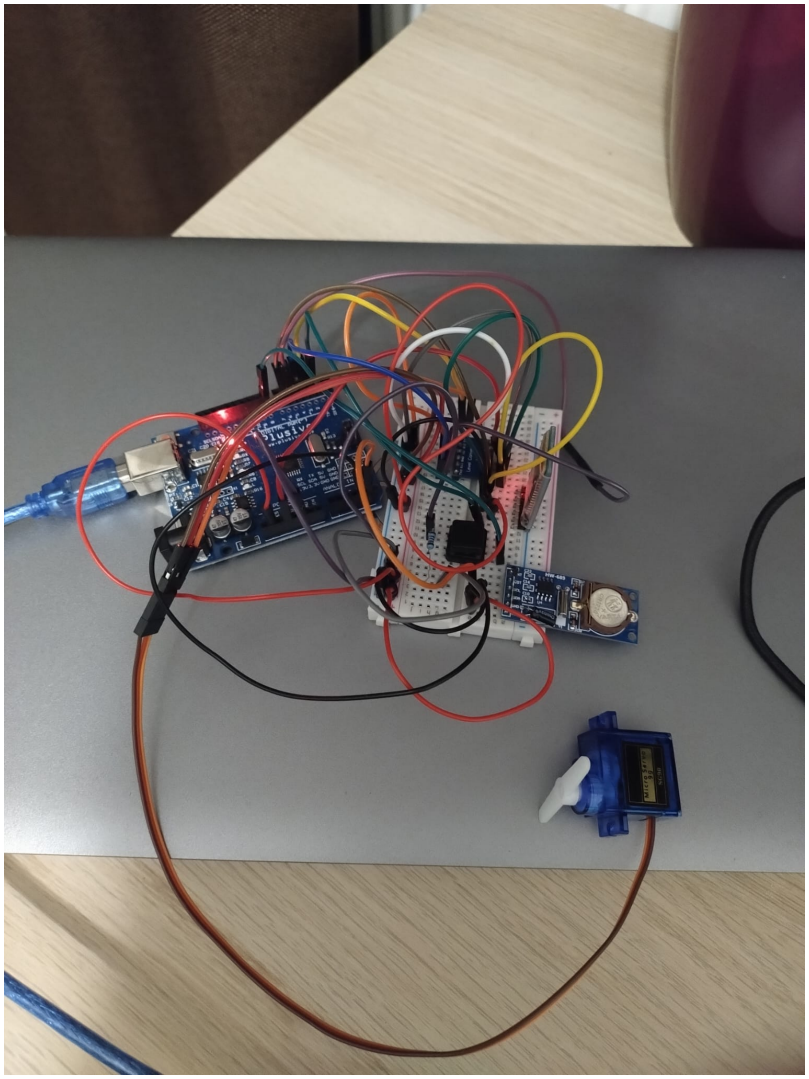
Apoi am conectat modulul Bluetooth, alimentat la 5V, dar liniile de date trec printr-un level converter care le duce la 3.3V, intrucat, conform spec sheet-ului pentru HC-05, inputul maxim acceptat pe liniile de date este de 4.2V, iar output-ul minim al pin-ului digital pe Arduino este de 4.1V; diferenta fiind prea mica exista riscul sa se arda modulul fara aceasta adaptare. Alternativ, puteam folosi un divizor format din 2 rezistori doar pentru linia de RX, intrucat TX-ul nu primeste semnal de la placuta, dar am ales sa folosesc o metoda mai integrata. Apoi am implementat o interfata seriala software, unde m-am lovit de un alt conflict: SoftwareSerial se foloseste de toate intreruperile disponibile (pe PORTB, C si D), chiar daca nu are nevoie neaparata de toate. Solutia a fost sa modific codul in SoftwareSerial.h sa folosesc doar intreruperea pe PCINT2, corespunzatoare portului PORTD unde este

conectata placuta. Apoi am alimentat pinul EN al HC-05 la 3.3V pentru a intra in modul AT, pentru a configura anumiti parametri ai modulului (baud rate mai mare, numele, slave mode, etc.). Dupa asta am deconectat pinul EN, intrucat nu mai aveam nevoie de el, si am testat transmisia bluetooth cu ajutorul unei aplicatii simple de bluetooth serial monitor.

In final, am conectat modulul RTC si am setat data si ora la cea curenta calculatorului, dupa care am integrat functia de TOTP care genereaza coduri bazat pe timestamp-ul curent dat de RTC, si o cheie secreta partajata cu aplicatia pentru telefon.

Pentru design-ul aplicatiei de telefon, am folosit Flutter, intrucat aveam putina experienta in acest SDK, pentru care am cautat biblioteci externe de bluetooth pentru comunicare seriala si de generare de TOTP, ca sa evit sa rescriu de la 0 generarea de hash-uri SHA1 si ulterior de HMAC-uri. Aici am dat de o problema: codurile generate de telefon si cele de placuta erau diferite, cu toate ca timestamp-urile si cheia secreta erau la fel, si generarea de hash-uri si de HMAC parea sa fie identica de asemenea, intrucat amandoua presupuneau compliance cu RFC 6238. Dupa ore intregi de chin, am remarcat ca biblioteca de TOTP pentru flutter face un padding in plus cheii secrete pana la 30 de octeti, pentru compliance cu codurile generate de google authenticator, care cere chei de dimensiune minima 30. Eliminand acest padding, codurile in sfarsit au coincis, si aplicatia functioneaza cum trebuie.

Rezultate Obţinute



Concluzii

Inainte de acest proiect, am avut 0 contact cu arduino si programare pe C embedded, iar in urma realizarii acestuia, am invatat mai multe si am realizat ca nu este atat de complicat pe cat credeam, iar utilitatea practica a proiectului (vreau sa-l pun la usa de la intrare :D) a facut lucrurile si mai interesante si placute.

Download

Link arhiva surse:
https://drive.google.com/file/d/1zYaWgyCK9QyTbBKiw8wBbDPN34eu805F/view?usp=share_link

Jurnal

01.05: Alegere tema proiect
07.05: Creare pagina wiki
14.05: Hardware design complet
21.05: Schema electrica completa
28.05: Aplicatie realizata complet + documentatie

Export to PDF