System kontroli obecności na podstawie elektronicznych legitymacji studenckich

(Students attendence verifing system with the use of electronic student's card)

Dawid Szczyrk

Praca magisterska

Promotor: dr Jakub Michaliszyn

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

6 września 2020

Streszczenie

Projekt wykonany w ramach tego opracowania, przedstawia realizację narzędzia umożliwiającego weryfikację uczestników wykładu na podstawie Elektronicznej Legitymacji Studenckiej.

Poniższa praca referuje proces od analizy problemu i postawienia wymagań do realizacji finalnej wersji wielokomponentowego projektu. Na efekt ostateczny składają się aplikacje wykorzystujące technologie webowe, oparte na popularnych frameworkach języka PHP oraz niskopoziomowe elektroniczne urządzenie zaprojektowane i wykonane na potrzeby tego projektu z wykorzystaniem platformy Arduino.

Pierwszym z elementów składających się na finalne rozwiązanie jest urządzenie elektroniczne - weryfikator, udostępniające zrozumiały interfejs użytkownika i realizujące zadania związane ze zbieraniem danych od uczestników wykładu. Zostało ono wykonane w oparciu o płytkę Arduino Uno rozszerzoną o dodatkowe komponenty w tym czytnik RFID i USB Host Shield. Całość układu, wraz z interfejsem do komunikacji z użytkownikiem, została połączona przy pomocy płytki stykowej i opakowana w sposób umożliwiający sprawne użytkowanie oraz wygodny transport.

Dodatkowo w ramach referowanego projektu wykonano internetową aplikację umożliwiającą katalogowanie zebranych przy pomocy weryfikatora obecności. Użytkownikowi umożliwiono zarejestrowanie indywidualnego konta w serwisie, w którego kontekście możliwe jest tworzenie struktury wykładów i zajęć odpowiadającej rzeczywistemu kalendarzowi prowadzącego. Aplikacja została wykonana przy użyciu frameworka języka PHP - Yii2. Została zaprojektowana w sposób niezależny od bazy danych studentów, aby umożliwić wygodne podłączenie systemu do istniejącego serwisu uczelnianego pozwalającego powiązać dane zebrane z ELS z informacjami o posiadającym ją studencie. Podczas pracy nad projektem nie było możliwości podłączenia się do żadnego istniejącego systemu uczelnianego udostępniającego dane studentów. Z tego powodu, jako element projektu, została stworzona aplikacja - w oparciu o framework Symfony4 - umożliwiająca pobranie danych studenta i udostępniająca w tym celu RESTowe Api. Obie stworzone aplikacje wraz z połączonymi z nimi bazami danych osadzone zostały w chmurze w sposób zapewniający bezpieczeństwo i niezawodność.

Within the following project, a student attendance verification system with usage of Electronic Students Card (ESC) was developed.

The presented work describes a process from problem analysis and demands defining to developing the final version of a multi-component project. The final system consists of Internet applications for collecting and managing studends presence data, and an electronic device made for this project with the usage of the Arduino platform.

The electronic device - the verificator - provides a comprehensive user interface and is responsible for collecting data about student presences. The final circuit was made based on the Arduino Uno board, extended with plenty of additional parts such as an RFID module and USB Host Shield. All interconnected parts were put in comfortable for use and transportation box.

For managing data collected by the verificator, an internet application was developed. It allows a user to create an account, and manage presences data in its context. It is possible to create a composition of lectures and classes, which corresponds with the real calendar of a lecturer. It was developed with the usage of the PHP framework - Yii2.
It was designed not to relate to concrete student's data managing system, to be flexible about a university system it is connected with.

Additionally, a system that allows correlation of data collected by verifier from ESC was developed. It takes the role of student data managing system, and provides REST API, to retrieve demanded information. It was created based on the PHP framework - Symfony4.

Both created services along with date bases connected with each of them were deployed in the cloud system with respect to proper security and reliability standards.

Spis treści

Rozdział 1.

Wprowadzenie

Zgodnie z ustępem 3 paragrafu 2 zarządzenia nr 116/2020 rektora Uniwersytetu Wrocławskiego [8] podczas wszystkich zajęć na uczelni, które odbywają się w formie stacjonarnej lub hybrydowej obligatoryjne jest prowadzenie listy obecności uczestników zajęć.

Panująca na świecie pandemia COVID-19 zmusiła ludzi do porzucenia wielu do tej pory dobrze sprawdzających się rozwiązań na rzecz takich, które pomogą zapobiec rozprzestrzenianiu się wirusa. Gospodarki i codzienne życie ludzi zostały przestawione na tryby tzw. nowej normalności.

W niemniejszym stopniu obecna sytuacja dotknęła pracy uniwersytetów. Po okresie zdalnego nauczania, wraz z nowym rokiem akademickim tradycyjne formy nauczania wracają do sal wykładowych, a troska o bezpieczeństwo studentów i pracowników uniwersytetu narzuca reorganizację życia akademickiego.

Jednym z elementów funkcjonowania instytucji naukowych, który powinien zostać dostosowany do nowych wymogów sanitarnych, jest prowadzenie listy obecności na zajęciach stacjonarnych. Obecna forma realizowana poprzez podawanie sobie przez studentów kartki na której spisane zostają ich dane prowokuje dodatkowe niebezpieczeństwo zakażenia wirusem.

Dodatkowym motywatorem, zachęcającym do usprawnienia obecnego rozwiązania, jest cytowane na początku zarządzenie rektora Uniwersytetu Wrocławskiej ustalające, że od semestru zimowego 2020 na wszystkich prowadzonych zajęciach obowiązkowe jest zbieranie listy obecności.

Względy zdrowotne, nie są jedynym uzasadnieniem konieczności optymalizacji identyfikowania osób biorących udział w zajęciach. Obecna forma tego procesu jest uciążliwa, zajmuje kilka pierwszych minut zajęć, a studenci się rozpraszają podając sobie listę obecności. Istotną kwestią którą należy wziąć pod uwagę jest również ochrona danych osobowych, do której w dzisiejszym świecie przykłada się coraz większą wagę.

W dobie współczesnych zdobyczy technologicznych można uznać za zaskakujące, że wciąż trwamy przy tym siermiężnym procesie. Wydawać by się mogło, że

obecne podejście jest tak zakorzenione w akademickiej kulturze, że nie ma możliwości go zmienić. Istnieją jednak opracowania które wskazują, że warto jest wdrażać nowoczesne rozwiązania, ponieważ ma to bezpośrednie przełożenie na prędkość sprawdzania obecności [6]. Zostało dodatkowo sprawdzone [1], że obecność ma wprost proporcjonalny wpływ na wyniki w nauce i odsetek uczniów porzucających studia przed ich ukończeniem. Wskazane zostało, że sprawdzanie obecności ma znaczenie nie tylko jako forma przymusu, ale również jako wyraz troski uczelni o swoich studentów i ma wpływ na ich morale

Rozdział 2.

Analiza problemu

2.1. Przegląd obecnego sposobu sprawdzania obecności

W celu doprecyzowania wymagań stawianych przed projektowanym przeze mnie system postanowiłem rozważyć znany mi z autopsji proces weryfikowania obecności na zajęciach prowadzonych przez Uniwersytet Wrocławski

Sprawdzanie listy obecności jest w dużej mierze uzależnione od preferencji prowadzącego i zwyczajów panujących na konkretnym wydziale - przykładowo obecność może być potwierdzona słownie - poprzez kolejne wywołanie osób z listy albo pisemnie - poprzez udostępnienie listy na którą uczestnicy będą zobowiązani się wpisać.

Dla uproszczenia dalszych rozważań postanowiłem przyjąć za ogólnie stosowaną metodę pisemną - która według moich obserwacji jest stosowana częściej - i na tej podstawie wyodrębnić czynności które składają się na wypełnienie listy obecności podczas zajęć.

Zbieranie informacji o obecności studentów na wykładzie jest czynnością na pierwszy rzut oka oczywistą, po bliższym przyjrzeniu się całej procedurze, okazuje się jednak, że daje się ona rozłożyć na jeszcze prostsze elementy.

- Przygotowanie kartki z listą obecności (przypisanie listy obecności do zajęć których będzie ona dotyczyła).
- Udostępnienie kartki uczestnikom oraz wpisywanie się na listę.
- Weryfikacja poprawności zebranej listy poprzez przeliczenie osób obecnych na zajęciach.
- Przeniesienie zebranych danych do uczelnianego systemu.

Powyższy schemat zbierania listy obecności podatny jest na dodatkowe błędy. Nie trudno wyobrazić sobie sytuację w której student intencjonalnie podał nie swoje

dane w celu zaliczenia obecności innej osobie. Kolejnym krokiem który może doprowadzić do błędów na ostatecznej liście obecności jest błąd ludzki podczas przenoszenia danych z kartki do uczelnianego systemu komputerowego. Taki błąd trudno jest od razu wychwycić ze względu na to, że studenci zazwyczaj nie przeglądają na bieżąco list obecności dostępnych np. na USOSie. Kiedy na koniec semestru obecność jest rozliczana bardzo trudno dojść do faktów.

Pierwszym elementem nadającym się w moim mniemaniu do usprawnienia jest samo tworzenie listy obecności z informację na temat zajęć których taka lista będzie dotyczyła. Fizyczna lista, poza czasem poświęconym na jej stworzenie, musi być po odbytych zajęciach przechowywana - co stwarza ryzyko jej zgubienia. Dodatkowo istnieje prawdopodobieństwo, że trafi w niepowołane ręce i ktoś nieżyczliwy wpisze na nią dodatkowe nazwiska. Do tego wszystkiego dochodzi jeszcze aspekt ekologiczny, ponieważ każda kolejna kartka generuje wymagające utylizacji śmieci.

Również udział studentów w rejestrowaniu własnej obecności może ulec optymalizacji. Wpisanie własnego imienia i nazwiska, a czasem numeru albumu zabiera czas, naraża na upublicznienie dane osobowe i utrudnia studentowi śledzenie wykładu.

Weryfikacja poprawności listy jest nie tylko pracochłonna, ale również podatna na błędy. Czas poświęcony na policzenie uczestników rośnie wprost proporcjonalnie do ich liczby. Dodatkowo prowadzący zajęcia może się zwyczajnie pomylić albo ktoś dopisać do listy już po tym sprawdzeniu.

Z całą pewnością usprawnienia wymaga proces przenoszenia danych z fizycznego nośnika do uczelnianego systemu komputerowego. Problemem jest nie tylko to, że podczas wpisywania obecności do komputerowego systemu można popełnić błąd, ale również to, że jest to proces czasochłonny.

2.2. Istniejące i możliwe realizacje zagadnienia

2.2.1. Podłączenie czytnika RFID bezpośrednio do komputera

Popularnym pomysłem usprawniającym system weryfikacji obecności jest skorzystanie technologii RFID (Radio-frequency identification). Wykorzystuje ona pole elektromagnetyczne do śledzenia na niewielkie odległości tagów (etykiet) RFID - małych transmiterów fal radiowych zdolnych do przechowywania danych umożliwiających zdalną identyfikację [7]. Tagi często mają formę kart, breloków lub innych łatwych w transporcie i przechowywaniu przedmiotów.

W opracowaniu [5] przedstawiono projekt weryfikatora obecności zrealizowany przy pomocy radio technologii. Każdemu studentowi został przypisany tag z unikatowym identyfikatorem. System składa się z czytnika etykiet podłączonego do komputera przy użyciu portu szeregowego oraz aplikacji komputerowej, zdolnej do rozpoznawania zbliżonych do czytnika tagów i zapamiętywania którzy studenci ze-

skanowali swoje karty w trakcie zajęć.

Ciekawym pomysłem przedstawionym w tym projekcie jest przekazanie osobnego tagu prowadzącemu zajęcia. Zbliżenie tej etykiety na początku i na końcu zajęć do czytnika wyznacza ramy czasowe podczas których obecność może zostać zaliczona studentom. Dodatkowo zastosowano tutaj podejście w którym uczestnik powinien rejestrować swoje pojawienie się jak i opuszczenie zajęć - jest to odstępstwo od tradycyjnej formy zbierania obecności.

W projekcie można zaobserwować kilka wad. Wykorzystany czytnik RFID jest urządzeniem w całości dostarczonym przez producenta. Można spodziewać się że jest to rozwiązanie drogie, a samego urządzenia nie będzie dało się rozszerzyć o dodatkowe funkcjonalności.

Taka architektura pociąga za sobą konieczność ciągłego połączenia z komputerem oraz dodatkowego okablowania. Może to powodować problemy logistyczne, jeśli chcielibyśmy korzystać z systemu w różnych salach.

Sama aplikacja komputerowa katalogująca zebrane obecności również poddana jest ograniczeniom. Korzystanie z tego systemu możliwe jest wyłącznie na komputerach, na których zainstalowano taką aplikację, co jest rozwiązaniem uciążliwym. Baza danych zapamiętująca obecności współdzielona jest pomiędzy poszczególne instancje aplikacji na różnych komputerach, jest więc ona udostępniona publicznie, co ułatwia nieautoryzowany dostęp.

2.2.2. Podłączenie przy pomocy kabla ethernetowego

W opracowaniu opisanym w pracy [5] zaproponowano podobną realizację, tym razem w oparciu o bezpośrednie podłączenie czytnika do internetu kablem RJ45. Wykorzystywane urządzenie jest stworzone na bazie Arduino Uno i rozszerzone o moduł RFID i gniazdo pozwalające podłączyć kabel ethernetowy.

Zaproponowany projekt rozwiązuje problem konieczności korzystania z komputera w bezpośrednim połączeniu z czytnikiem, a wybrany układ elektroniczny pozwala rozszerzyć urządzenie o dodatkowe funkcjonalności. Łączność internetowa pozwala na natychmiastowe odnotowywanie obecności w bazie, co na pewno jest wygodnym rozwiązaniem.

W treści opracowania nie został poruszony problem połączenia obecności z zajęciami na których została ona odnotowana. W przypadku rozszerzenia systemu o kolejne czytniki podłączone do sieci i korzystania z wielu urządzeń jednocześnie, nie będzie możliwości rozszyfrowania, z jakich zajęć pochodzą spływające do serwera dane.

Jedynym możliwym sposobem na komunikację urządzenia z użytkownikiem, jest w tym wypadku buzzer, co może okazać się niewystarczające do wyrażenia stanu urządzenia. W razie kłopotów z łącznością albo anteną RFID, nie ma możliwości wygodnego sygnalizowania problemu.

2.2.3. Samodzielny czytnik

W cytowanym we wstępie opracowaniu [6], w celu rozwiązania powyższego problemu zaproponowano wykorzystanie z wyświetlacza LCD, z którego użytek robi opisany w tej pracy układ elektroniczny. Przedstawione w pracy urządzenie poza czytnikiem RFID, zdolne jest również do wyświetlania treściwych komunikatów potwierdzających udane odnotowanie karty studenta.

Trudno w tym wypadku dyskutować jednak z innymi rozwiązaniami technicznymi zawartymi w tej pracy, ponieważ motywem przewodnim tego opracowania jest pokazanie, że sprawdzanie obecności przy pomocy tagów RFID przebiega szybciej niż w klasycznym podejściu.

2.2.4. Aplikacja internetowa umożliwiająca rejestrowanie obecności studentów na zajęciach przez internet lub telefon

W mojej ocenie to najprostsze rozwiązanie składające się z jednolitego systemu, w którym jednocześnie można zbierać obecności uczestników wykładu, oraz je przechowywać i przeglądać. Jej niewątpliwym plusem jest całkowite wyeliminowanie problemu związanego z wprowadzaniem danych do systemu katalogującego obecności, ponieważ dane wprowadzane są wprost do systemu. Identyfikacja konkretnych zajęć w ramach których rejestrowana jest obecność, odbywałaby się na początku zajęć poprzez wejście w kontekst odpowiedniego wykładu w aplikacji. Obecność uczestników na zajęciach byłaby sprawdzana przez prowadzącego i od razu wprowadzana do systemu.

Nie zdecydowałem się na to rozwiązanie, ponieważ wydaje mi się ono nieszczególnie odbiegać od aktualnej procedury, poza zmianą nośnika z kartki papieru na ekran tabletu.

2.2.5. Urządzenie rejestrujące obecności za pośrednictwem sieci bezprzewodowej

To rozwiązanie początkowo wydało mi się bardzo atrakcyjne i przez długi czas miałem zamiar je zrealizować. Planowałem rozszerzyć mikrokontroler o moduł do bezprzewodowej łączności z internetem i w stworzonej aplikacji udostępnić interfejs do odnotowywania obecności.

Po dłuższych rozważaniach okazało się jednak, że takie rozwiązanie przerasta moje możliwości. Ten sposób wysyłania danych do aplikacji katalogującego obecności wymagałby umożliwienia powiązania obecności z zajęciami, na których zostały zebrane

Wymagałoby to dołączenia dodatkowego interfejsu który pozwalałby wprowadzać takie dane, co jeszcze bardziej komplikowałoby projektowane przeze mnie urządzenie. Dodatkowo ograniczyłoby to zbiór układów scalonych z których mógłbym

skorzystać, ponieważ musiałbym wybrać takie, które posiadają wystarczającą ilość pamięci, pinów i innych potrzebnych peryferiów.

Ostatecznie moje plany dotyczące takiego projektu pokrzyżował moduł ESP, który miał realizować komunikację bezprzewodową, ale okazał się bardzo kapryśny.

2.3. Określenie wymagań wobec projektu

W poprzednim rozdziale wspomniano o wielu możliwych realizacjach problemu zbierania obecności, co jest i tak tylko niewielkim wycinkiem możliwych do wykonania projektów. W opracowaniu [4] wspomniano o wielu innych przedsięwzięciach, z których na wspomnienie zasługują na pewno:

- Skorzystania z telefonu komórkowego i łączności bluetooth w celu odnotowania obecności.
- Wykorzystanie odcisku palca do dodatkowego potwierdzenia tożsamości studenta.
- Użycia kamery i rozpoznawania twarzy w celu identyfikacji wchodzących na wykład studentów.

Analiza powyższych rozwiązań wraz z ich mocnymi i słabymi stronami pozwoliła wyłonić zbiór wymagań wobec projektowanego systemu. Wymagania zostały dobrane pod kątem poprawienia wydajności klasycznego sposobu weryfikowania obecności, co zostało rozważone w rozdziale 2.1, oraz możliwości jakie dają współczesne technologie - które zostały opisane powyżej.

Urządzenie powinno udostępniać następujące funkcjonalności':

1. Zarejestrowanie obecności uczestnika zajęć przy pomocy tagu RFID.
2. Wyświetlanie jasnych komunikatów odnośnie stanu urządzenia.
3. Powiązanie obecności z zajęciami, podczas których zostały zebrane.
4. Potwierdzenie tożsamości uczstnika odnotowującego swoją obecność.
5. Katalogowanie zebranych danych i generowanie raportów podsumowujących.
6. Wygodne przeniesienie obecności zebranych przez czytnik do aplikacji przechowującej dane.

Rozdział 3.

Określenie struktury projektu

Finalnie zdecydowałem, żeby jednoznacznie oddzielić sposób zbierania danych od ich przechowywania i przetwarzania.

W realizacji projektu skorzystałem z faktu, że polskie uczelnie powszechnie korzystają z Elektronicznych Legitymacji Studenckich (ELS), które zawierają wbudowany tag RFID.

Komponent umożliwiający studentowi odnotowanie swojej obecności na wykładzie - weryfikator - postanowiłem oprzeć o platformę Arduino - znaną mi z zajęć systemów wbudowanych. Urządzenie zostało rozszerzone o moduły umożliwiające realizację założeń z poprzedniego rozdziału.

Czytnik fal radiowych umożliwiający odczytanie danych z etykiety RFID znajdującej się na ELS. Gniazdo USB pozwalające podłączyć pendrive na którym zostaną przeniesione dane z urządzenia na bazie Arduino do aplikacji internetowej przechowującej dane.

Dodatkowo częścią urządzenia będzie zegar RTC umożliwiający zapamiętanie godziny, o której odnotowane zostały obecności, a więc powiązanie ich z konkretnymi zajęciami, oraz wyświetlacz LCD + Buzzer, pozwalające na wygodne korzystanie z samodzielnego urządzenia.

Sposobem na potwierdzenie tożsamości studenta, będzie w wypadku realizowanego projektu porównanie zdjęcia na legitymacji z twarzą osoby która zbliża ją do czytnika. Za tę weryfikację odpowiedzialny będzie prowadzący.

Rolę oprogramowania odpowiedzialnego za katalogowanie zebranych przez weryfikator danych będzie pełnić w moim systemie ogólnodostępna aplikacja internetowa, umożliwiająca założenie konta i zarządzanie zebranymi obecnościami w kontekście całego przedmiotu i w kontekście pojedynczych zajęć. Dodatkowo ma ona umożliwiać generowanie raportów ze zagregowanymi danymi, tak aby dało się łatwo prześledzić obecności w kontekście całego semestru.

Postanowiłem uczynić realizowany projekt niezależnym od danych studentów.

Dzięki takiemu podejściu w łatwy sposób będę mógł przełączyć cały system na korzystanie z baz danych na różnych uczelniach oraz uniknę konieczności przechowywania wrażliwych danych w systemie.

Ze względu na to, że na etapie realizowania założonego projektu nie uzyskałem dostępu do serwisu udostępniającego dane wrocławskich studentów, stworzyłem własną, niezależną od innych części projektu, aplikację realizującą funkcję bazy danych studentów.

Umożliwia ona powiązanie danych pobranych z legitymacji studenckiej z danymi pozwalającymi na jego identyfikację. Jest to aplikacja rozłączna względem systemu zarządzającego obecnościami osób uczestniczących w zajęciach, wystawiająca RESTowe api.

Rozdział 4.

Realizacja

4.1. Weryfikator - czytnik legitymacji studenckich

Zgodnie z wymaganiami projektowane przeze mnie urządzenie powinno udostępniać co najmniej następujące możliwości.

- Możliwość podłączenia zewnętrznego nośnika danych.
- Stworzenie pliku na którym zapamiętani zostaną uczestnicy zajęć
- Zebranie informacji z elektronicznej legitymacji studenckiej, potwierdzające obecność uczestników
- Wyświetlanie informacji o stanie urządzenia, w szczególności sukcesu odczytania danych z legitymacji
- Zapamiętanie czasu kiedy zebrane zostały obecności

Dodatkowo, chciałbym żeby cała elektronika schowana była wewnątrz plastikowego opakowania, a zasilanie dostarczane za pomocą kabla podłączonego do sieci elektrycznej. Opakowanie powinno zostać wyposażone w otwory na kabel zasilający, zewnętrzny nośnik danych, wyświetlacz. Czytnik kart RFID powinien być tak zlokalizowany aby umożliwić łatwe zebranie danych z ELS.

4.1.1. Zastosowana platforma i peryferia

Arduino to cała platforma programistyczna umożliwiająca łatwe projektowanie i testowanie układów składających się z wielu elektronicznych modułów. Podstawą każdego projektu Arduino jest układ elektroniczny oparty na mikroprocesorze. Pod egidą tego systemu udostępniony jest cały wachlarz wersji takich układów, różniących się rodzajem kontrolera i udostępnionymi peryferiami (przykładowo: slot na kartę SD, gniazdo RJ-45, wbudowany wyświetlacz, czy moduł WIFI). Do wyboru

użytkownika pozostaje cała gama wersji płytek drukowanych spod znaku Arduino. Dodatkowo platforma udostępniona jest na licencji open hardware, co sprawiło, że z projektem związana jest olbrzymia międzynarodowa społeczność entuzjastów elektroniki. Częścią platformy jest również środowisko programistyczne umożliwiające wgrywanie własnych programów na podłączoną do komputera płytkę.

Ciekawą opcją dostępną dla użytkowników platformy jest możliwość skonstruowania własnej płytki z procesorem na bazie powszechnie dostępnych schematów gotowych układów. Może to być rozwiązanie bardzo atrakcyjne w przypadku kiedy projektantowi zależy na ograniczeniu kosztów układu. W moim rozwiązaniu postanowiłem nie iść tak daleko i skorzystać z jednej z gotowych płytek.

Podstawą mojego weryfikatora postanowiłem uczynić płytkę Arduino Uno oparte na mikroprocesorze AVR ATmega328. Kontroler w tym układzie jest bardzo prosty, posiada bardzo mocno ograniczone zasoby, co wymusza dodatkowy wysiłek włożony w zarządzanie pamięcią i czasem procesora. Do mojej dyspozycji miałem następujące zasoby

- 32kB Flash - przestrzeń na kod programu
- 2kB SRAM zajmowane przez pamięć operacyjną.
- porty SPI, I2C, szeregowy
- Jednostka obliczeniowa o częstotliwości zegara 20MHz

Dodatkowo, w celu zapewnienia wszystkich wymaganych funkcjonalności przez projektowany przeze mnie weryfikator, skorzystałem z następujących dodatkowych modułów:

- Czytnik RFID - do odczytywania danych z legitymacji studenckiej wyposażonej w interfejs zbliżeniowy. Czytnik podłączony jest przez port szeregowy
- USB Host Shield - dodające gniazdo USB, umożliwiające podłączenie pendriva, korzystający z drugiego portu SPI
- Wyświetlacz LCD wraz z potencjometrem do ustawiania kontrastu, przekazujący użytkownikowi informacje na temat stanu urządzenia
- RTC - zegar czasu rzeczywistego z baterią do zapamiętywania czasu zbierania danych, korzystający z interfejstu I2C
- Buzzer piezo wydający dźwiękowe sygnały w razie sukcesu

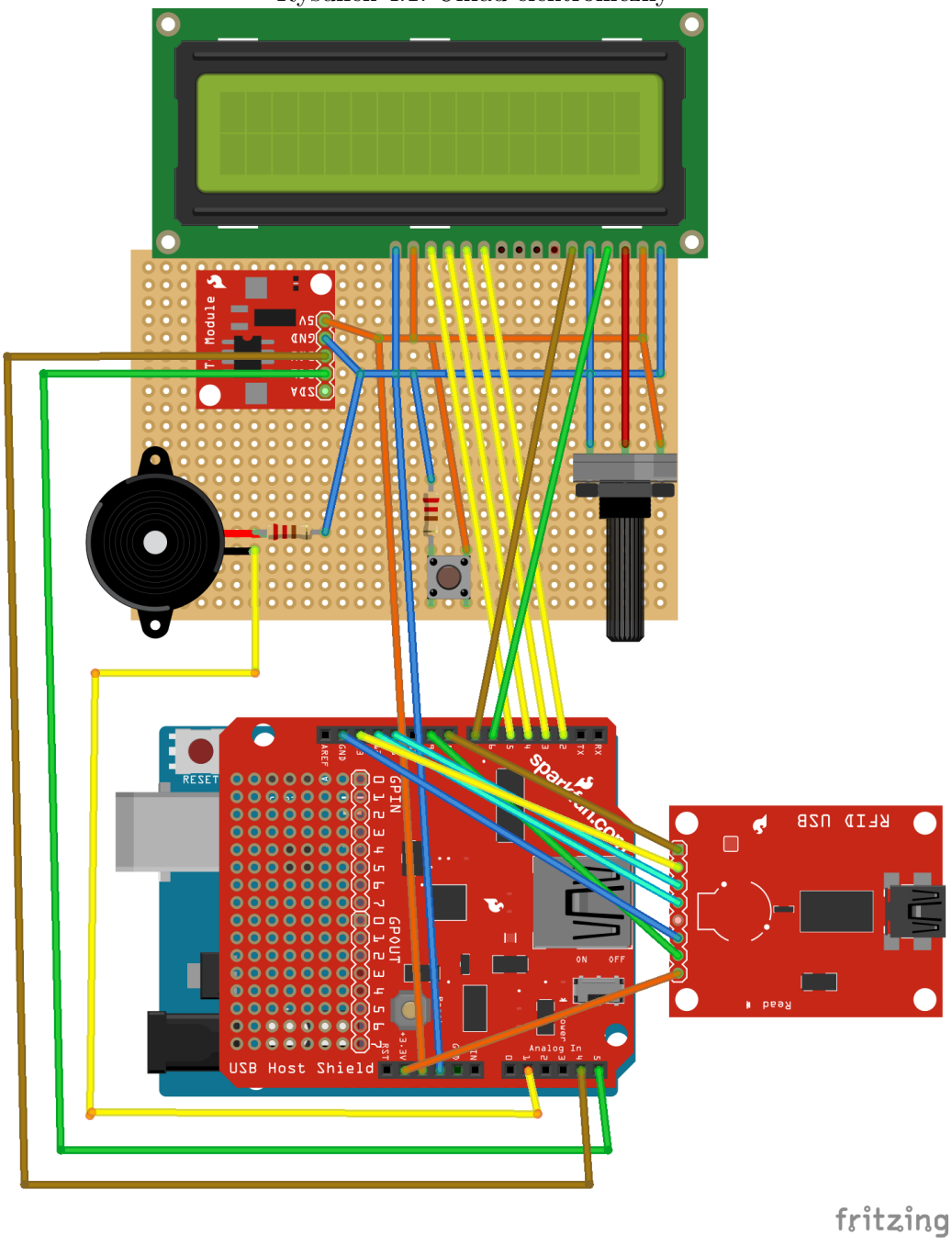
4.1.2. Konstrukcja

Do skonstruowania ostatecznego układu z zebranych elementów używam dodatkowo:

- Jednej dwustronnej płytki stykowej
- Dwóch rezystorów 220 Ohm
- kilkudziesięciu kabli różnej długości
- Jednego przycisku do resetowania zegara

Cały układ został osadzony w plastikowym opakowaniu z wywierconymi otworami

Rysunek 4.1: Układ elektroniczny



4.1.3. Działanie

Osią działania zaprojektowanego przeze mnie weryfikatora jest interakcja z dwoma zewnętrznymi urządzeniami. Jednym z nich jest nośnik danych na którym zapisane zostaną obecności uczestników zajęć. W realizowanym projekcie rolę tę pełni pendrive sformatowany w systemie FAT. W zależności od tego czy w danym momencie pendrive jest podłączony, urządzenie jest w stanie oczekiwania na drugie z zewnętrznych narzędzi - Elektroniczną Legitymację Studencką. Dodatkowo weryfikator jest zdolny do wyświetlanie komunikatów o swoim stanie na ekranie LCD i wydawania sygnałów dźwiękowych przy pomocy brzęczka. Częścią urządzenia jest również zegar RTC, który pozwala śledzić aktualną godzinę. Weryfikator monitoruje swój własny stan i jest zdolny do wykrywania błędów takich jak:

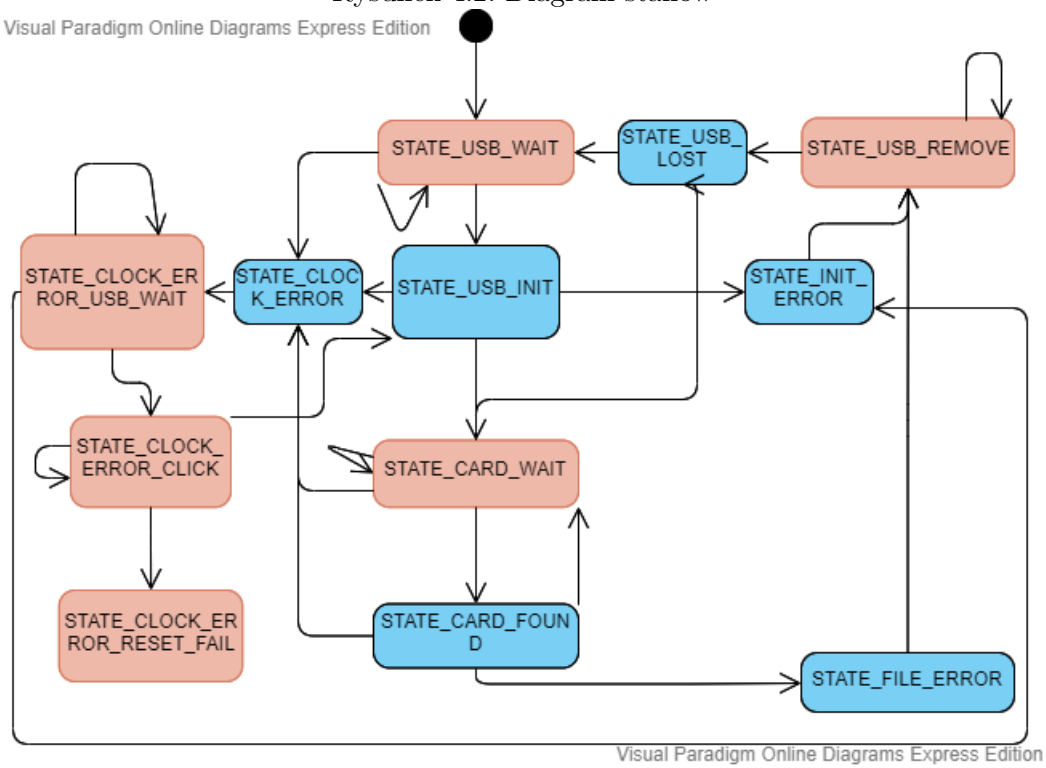
- Błąd Inicjalizacji podłączonego pendrive'a
- Błąd zapisu odczytanych danych na nośniku
- Błąd zegara RTC - spowodowanego najczęściej rozładowaną baterią

Kod mojego weryfikatora projektowałem z myślą o tym, żeby łatwo dało się przedstawić przebieg programu na diagramie. Struktura kodu jest przemyślana w taki sposób, żeby podczas jednego przebiegu pętli głównej wykonywał się jedynie kod powiązany z aktualnym stanem urządzenia. Taki układ nie tylko pozwala zwięźle opowiadać o działaniu urządzenia, ale również ułatwia testowanie poprawności podczas pisania kodu i montowania urządzenia.

Weryfikator w danej chwili może znajdować się w jednym z 12 stanów. Na przedstawionym diagramie stany oczekujące na zdarzenie przedstawiono kolorem kremowym, a stany przejściowe kolorem niebieskim:

- STATE_USB_WAIT - oczekiwanie na pendrive - ekran wyświetla informację - "Insert pendrive". Próba inicjalizacji nośnika zajmuje około 5 sekund dlatego wyświetlany czas jest aktualizowany z opóźnieniem
- STATE_USB_INIT - inicjalizacja pendriva - po odnalezieniu pendriva wyświetlana jest informacja "USB Inserted". W przypadku błędu inicjalizacji - przejście do stanu STATE_INIT_ERROR, w razie sukcesu na podstawie aktualnego czasu stworzona i zapamiętana zostaje nazwa YY-MM-DD-hh_mm_ss dla pliku który będzie przechowywał zebrane uidy. STATE_CARD_WAIT.
- STATE_CARD_WAIT - oczekiwanie na kartę RFID - ekran: Ścan Card". W razie wykrycia odłączenia pendriva przejście do stanu STATE_USB_LOST. W przypadku wykrycia karty, ale błędu w odczycie (np. ze względu na zbyt szybkie odsunięcie karty) wyświetlenie informacji: Read error.
- STATE_CARD_FOUND - zapisywanie Uidu do pliku - otwarcie pliku o zapamiętanej nazwie zapisanie do niego odczytanego Uidu. W razie błędu przy

Rysunek 4.2: Diagram stanów



zapisie przejście do stanu STATE_FILE_ERROR, w przypadku sukcesu sygnał dźwiękowy, wyświetlenie informacji Żead Success"i powrót do stanu STATE_CARD_WAIT

- STATE_INIT_ERROR - błąd inicjalizacji pendriva - ekran: "Init Error"
- STATE_USB_REMOVE - czekanie na odłączenie pendriva. Wyświetlacz:"Reinsert USB". W razie wykrycia odłączenia pendriva przejście do stanu STATE_USB_LOST.
- STATE_USB_LOST - odłączono pendrive . Ekran: "USB removed", sygnał dźwiękowy oraz przejście do stanu STATE_USB_WAIT
- STATE_FILE_ERROR - błąd zapisu do pliku. Ekran: "Open file error", przejście do stanu STATE_USB_REMOVE
- STATE_CLOCK_ERROR - błąd zegara - w razie wykrycia problemów z zegarem w którymś ze stanów [STATE_USB_WAIT, STATE_CARD_WAIT, STATE_CARD_FOUND, STATE_USB_INIT, STATE_USB_REMOVE] sterowanie przechodzi do tego stanu. Wyświetlona zostaje informacja: Ćlock Error", następnie sterowanie przechodzi do STATE_CLOCK_ERROR_USB_WAIT
- STATE_CLOCK_ERROR_USB_WAIT - oczekiwanie na pendrive z czasem - wyświetlacz "Insert pendrive with time". Sterowanie pozostaje w tym stanie do czasu kiedy na zewnętrznym nośniku wykryty zostanie plik o nazwie tdav, który w pierwszym wierszu będzie zawierał datę w formacie "YYYY/MM/DD hh:mm:ss". W przypadku wykrycia poprawnego pliku program zapamiętuje

datę z pliku tdav i przechodzi w stan STATE_CLOCK_ERROR_CLICK
- W razie błędu przy inicjalizacji pendriva program przechodzi do stanu STATE_INIT_ERROR
- W razie wykrycia pendrive i nie wykrycia pliku o nazwie tdav wyświetlona zostaje informacja "Upload time tdav"
- W razie złego formatu pierwszej linijki w pliku tdav wyświetlona zostaje informacja: "Bad date format"

- STATE_CLOCK_ERROR_CLICK - oczekiwanie na reset. Wyświetlacz: Ćlick to reset". Sterowanie pozostaje w tym stanie do czasu kiedy zostanie przyciśnięty guzik resetu. Po naciśnięciu guzika na zegarze zostaje ustawiona zapamiętana data, wyświetlony zostaje komunikat "Time set", a sterowanie przechodzi w stan STATE_USB_INIT. Jeśli podczas ustawiania czasu na zagarze wystąpi błąd program przechodzi w stan STATE_CLOCK_ERROR_RESET_FAIL

- STATE_CLOCK_ERROR_RESET_FAIL - błąd resetowania czasu. Wyświetlacz: "Clock Reset Fail". Ze względu na brak możliwości ustawienia poprawnego czasu jest to stan, z którego urządzenie nie może wrócić do poprawnego funkcjonowania.

4.1.4. Procedura resetowania zegara RTC

4.1.5. Napotkane problemy

Dwa moduły korzystające z interfejsu SPI

Początko zamiast Arduino Uno planowałem użyć innej wersji tej popularnej płytki - Arduino Leonardo.

Tak jak w wersji którą ostatecznie zrealizowałem, planowałem do układu Leonardo podłączyć USB Host Shield oraz moduł RFID - oba używają interfejsu SPI. Każdy z tych komponentów osobno współpracował poprawnie z Arduino Leonardo. Problemy pojawiły się kiedy próbowałem podłączyć wszystko w jedną całość i zapisywać odczytane dane z karty magnetycznej na podłączonym pendrivie.

Podczas montowania układu problemem nie do przeskoczenia okazało się to, że w przypadku tej wersji płytki piny interfejsu SPI znajdują się w magistrali ICSP, która jest zajęta przez moduł USB w taki sposób, że nie ma możliwości podłączyć tam innych kabli.

Próbowałem jeszcze dolutować kable do magistrali CSP na płytce USB Host Shield, jednak ze względu na to że, są to jedynie milimetrowe wypustki, przepaliłem któreś z połączeń wewnątrz tego układu i ostatecznie musiałem je wyrzucić.

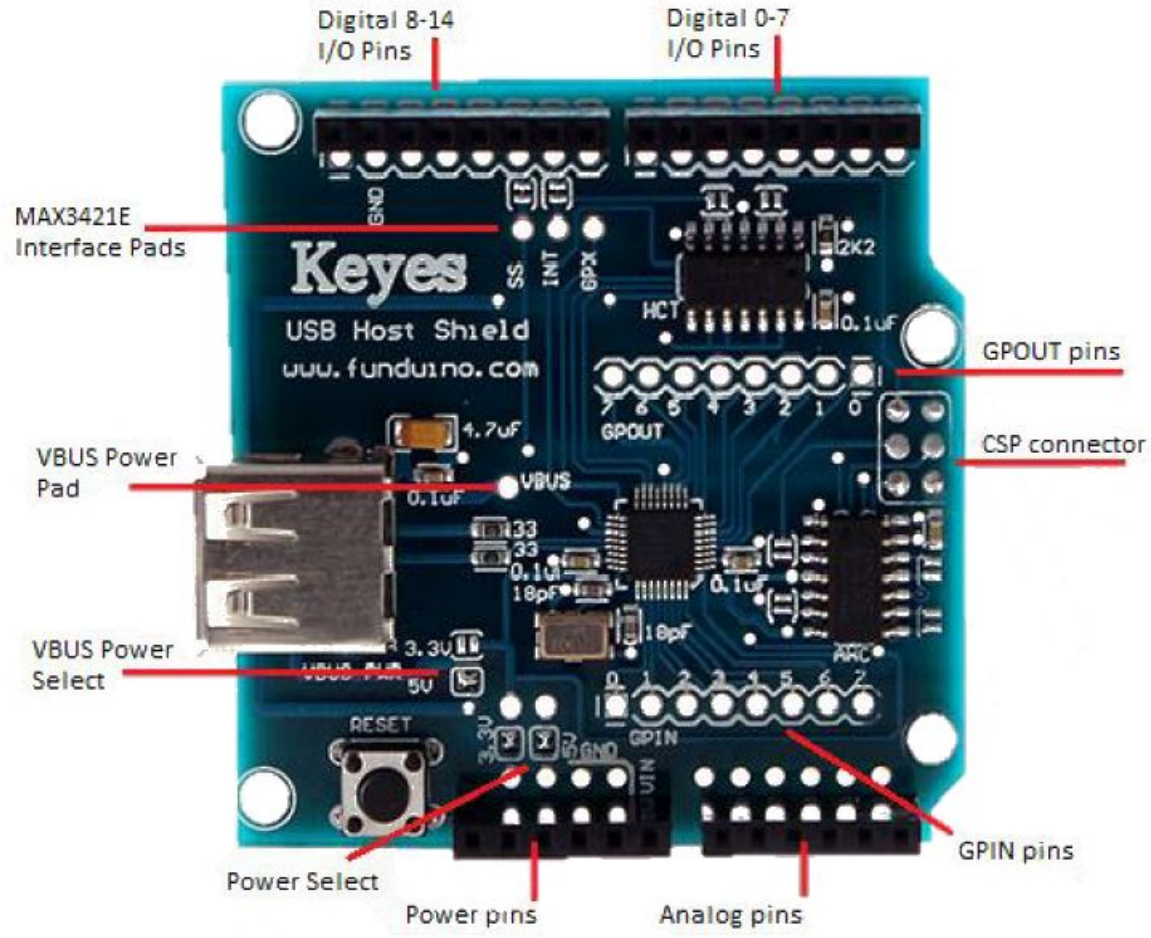
W celu rozwiązania problemu z portem SPI przeszedłem na wersję UNO które interfejs SPI ma doprowadzony również na piny 10-13.

Problemy z odczytaniem danych z ELS

W początkowej wersji urządzenie korzystało z modułu RFID - MFRC-522. Ten komponent bez zarzutów współpracował z załączonymi w zestawie tagami, jednak bardzo słabo wykrywał Elektroniczną Legitymację Studencką. Odczytanie Uidu z ELS zajmowało zazwyczaj parę minut. Żeby rozwiązać ten problem postanowiłem zamienić moduł RFID, na taki o którym czytałem, że ma lepszy zasięg - PNS532. Pierwsze próby zwiastowały pełny sukces - próby odczytania danych z ELS przebiegały bez zarzutu.

Problem pojawił się jednak przy współpracy nowego modułu z resztą układu. Jak już zostało wcześniej wspomniane, Arduino Uno posiada jeden port SPI, do którego można podłączyć więcej urządzeń. Niestety w tym wypadku rzeczywistość okazała się odbiegać od oczekiwań. Arduino było w stanie współpracować tylko z jednym z dwóch komponentów - albo USB Host shield albo modułem PNS532. Ostatecznie okazało się jednak, że moduł PNS532 udostępnia również inne porty, dzięki czemu udało mi się go podłączyć przy pomocy portu szeregowego.

Rysunek 4.3: Uklad USB Host shield ze wskzaniem na zakończenia magistrali CSP



Wypełniona pamięć operacyjna

W miarę dołączania kolejnych komponentów do układu, tzn. zegara RTC, brzęczka oraz wyświetlacza LCD, zacząłem natrafiać na problemy z pamięcią.

Pierwsza skończyła się pamięć operacyjna której mój kontroler posiadał jedynie 2KB. Objawiało się to nagłym zawieszaniem się wykonywania programu, co w pierwszym momencie było dla mnie zaskakujące i niezrozumiałe.

Udało mi się rozwiązać ten problem dzięki większej dyscyplinie w korzystaniu z pamięci. Ograniczyłem liczbę zmiennych globalnych, a największy efekt przyniosło przeniesienie wykorzystywanych w programie napisów z pamięci operacyjnej do pamięci flash, do czego służy dyrektywa PROGMEM.

Pamięć flash jest przeznaczona do przechowywania kodu programu i nie ma do niej swobodnego dostępu jak w przypadku pamięci operacyjnej. Dodanie kolejnej biblioteki która ułatwiłaby przechowywanie danych w tej przestrzeni nie było możliwym ze względu na ograniczone zasoby.

Ostatecznie zdecydowałem się na korzystanie z pamięci flash wczytując dane bajt po bajcie. Chcąc więc wyświetlić komunikat na ekranie LCD, tak czy inaczej musiałem najpierw wczytać bajtowo łańcuch znaków do pamięci operacyjnej, jednak nie było już potrzeby przechowywać wszystkich łańcuchów znaków jednocześnie w SRAM.

Rysunek 4.4: Stałe programu przechowywane w pamięci flash

```
//Deklaracja łańcucha znaków w pamięci operacyjne
const char LCD_INSERT[] PROGMEM = "Insert pendrive";

//Deklaracja łańcucha znakóW w pamięci flash
const char LCD_INSERT[] = "Insert pendrive";

//Ładowanie stałej z pamięci flash do pamięci operacyjnej
// const_ptr - wskazuje na początek łańcucha znakó w pamięci flash
for (byte aux_byte = 0; aux_byte < strlen_P(const_ptr); aux_byte++) {
    lcd.print((char)pgm_read_byte_near(const_ptr + aux_byte));
}
```

Wypełniona pamięć flash

Przeniesienie łańcuchów znaków do pamięci flash, chociaż poprawiło wydajność pamięci operacyjnej, to spowodowało wyczerpanie przestrzeni przeznaczonej na kod programu.

Pamięć flash jest bardziej pojemna niż SRAM ponieważ w tym wypadku jest to aż 32KB do dyspozycji programisty, jednak duża liczba bibliotek do obsługi peryferiów doprowadziła do wyczerpanie się również tej przestrzeni.

Ułatwieniem w przypadku optymalizowania wykorzystania pamięci flash jest to, że przestrzeń zajmowana przez kod programu jest stała po kompilacji. Nie ma więc obawy że w trakcie przebiegu programu nastąpi nagły wzrost i przekroczenie

pojemności. Wystarczy dostosować się do limitu 32 KB.

Udało mi się tego dokonać poprzez ograniczenie liczby funkcji które deklarowałem w moim programie, oraz schowanie komunikatów wysyłanych na serial monitor wewnątrz dyrektyw preprocesora #if DEBUG serial.println("SETUP"); #endif. Dzięki temu mogłem łatwo wyłączyć kod programu odpowiedzialny za debugowanie.

Moduł WIFI

W pierwszej wersji planowanego przeze mnie weryfikatora, miał on się łączyć z siecią za pomocą modułu WIFI. Bardzo dużo kłopotów sprawił mi jednak układ ESP8266, odpowiedzialny za tę łączność. Po wgraniu sterownika do tego układu przestawał on być wykrywany przez komputer i Arduino.

Pomimo wykorzystania kilku takich układów za każdym razem pojawiał się ten sam problem. Nie udało mi się dociec jaka była tego przyczyna. Ostatecznie odszedłem od wykorzystania modułu ESP8266 ze względu na zmianę koncepcji.

4.1.6. Inne koncepcje weryfikatora

Zastanawiając się nad ostateczną formą jaką powinien przyjąć projektowany przeze mnie weryfikator brałem jeszcze pod uwagę układ Raspberry Pi. Ostatecznie zrezygnowałem z tego pomysłu, ze względu na to że takie urządzenie mogłoby wymagać utrzymywania systemu operacyjnego aktualnej wersji. Za wykorzystaniem Arduino przemawia również cena takiego układu i fakt, że jest on mi bardzo dobrze znany z zajęć Systemów Wbudowanych.

Projektując układ musiałem również podjąć decyzję z jakiego nośnika skorzystać do przenoszenia danych z weryfikatora do aplikacji zarządzającej obecnościami. Brałem pod uwagę dwie możliwości. Poza opcją na którą się ostatecznie zdecydowałem, brałem jeszcze pod uwagę korzystanie z kart SD

Ze względu na większą dostępność i łatwiejsze podłączenie klasycznego pendrive'a do komputera (nie każdy komputer posiada slot na kartę SD), zdecydowałem się nie tę właśnie możliwość. Później okazało się, że ta decyzja miała swoje konsekwencje.

Po pierwsze USB Host Shield zajął magistralę ICSP, będącą jedynym wyjściem pinów interfejsu SPI w układzie Arduino Leonrado. Po perypetiach wcześniej już przeze mnie opisanych, zdecydowałem się na zmianę wersji płytki na Uno.

Kolejnym utrudnieniem które zostało postawione na mojej drodze ze względu na korzystanie z pendriva, były problemy z dostępnością biblioteki obsługującej pamięć w tej formie. Podstawowa biblioteka obsługująca układ USB Host Shield nie przewiduje wykorzystania złącza USB do podłączenia pamięci masowej.

Ostatecznie udało mi się znaleźć nieszczególnie popularną bibliotekę do obsługi tego narzędzia - UsbFat, która jednak okazała się być słabo opisana.

4.2. Elektroniczny dziennik

W celu wygodnego przetwarzania zebranych danych o obecnościach stworzona została aplikacja internetowa - elektroniczny dziennik. W serrwisie użytkownik może założyć konto, w jego kotekście tworzyć strukturę prowadzonych przez siebie zajęć i uzupełniać listę obecnych na nich studenów .

Dziennik umożliwia bieżące przeglądanie obecności studentów uczesniczących w wykładzie oraz, w dowolnym momencie, wygenerowanie raportu w formacie CSV ze zagregowanymi obecnościami uczesników wykładu na zajęciach z których się on składa.

Struktura dziennika

Użytkownik serwisu po założeniu konta i zalogowaniu ukazuje się lista instancji wykładów, które są przypisane do jego konta i na których może dokonać standardowych operacji przeglądania/usuwania/dodawania. Instancja wykladu składa się z nazwy, opisu, listy uczesników oraz listy zajęć w jej ramach. Instancję wykladu można rozumieć jako zajęcia prowadzone przez jednego prowadzącego z grupą studentów w ramach jednego przedmiotu.

Do instancji wykładu zostaje przypisana grupa studentów, którzy są zapisani na zajęcia prowadzone przez użytkownika w ramach wykładu. Przykładową listą uczestników będzie więc jedna grupa studentów uczesniczących w ćwiczeniach prowadzonych przez jednego pracownika uniwerytetu w ramach jednego przedmiotu. Dodatkowo, generowany raport z obecności na jednej instancji wykładu będzie odnosił się tylko do studenów znajdujących się na tej liście.

Każda instancja wykładu posiada skończony zbiór zajęć, których pojedyncza instancja składa się z daty i godziny oraz przypisanej do niej listy obecności. Taka instancja może być rozumiana jako pojedyncze zajęcia jednej grupy studentów. Na listę obecności zajęć mogą zostać wpisani studenci, którzy nie są wpisani na listę uczesników instancji wykładu. Będą widoczni w przeglądzie konkretnych zajęć w graficznym interfejsie dziennika, jednak nie zostaną wzięci pod uwagę przy generowaniu raportu dla instancji wykładu.

Posługując się przykładem zajęć z AISDu prowadzonych w ramach pojedynczego semestru, struktura w dzienniku będzie wyglądała następująco: Przedmiot będzie rozdzielony pomiędzy konta wykładowcy i wszystkich ćwiczeniowców prowadzących zajęcia.

Wykladowca stworzy na swoim koncie w elektornicznym dzienniku instancję wykładu, przypisane do niego zostaną wszystkie wykłady, a listą uczesników będą wszyscy studenci zapisani na ten przedmiot. Lista obecności pojedynczych zajęć będzie więc składać się ze wszystkich studentów, którzy w trakcie odbywania się wykładu w sali Instytutu Informatyki przyłożą swoją legitymację do weryfikatora.

Ćwiczeniowcy na swoich kontach utworzą osobne instancje wykładu, przypiszą do nich zajęcia odpowiadające prowadzonym przez nich cotygodniowym ćwiczeniom oraz dodadzą grupę studentów, którzy zapisali się do ich grupy.

Pojedynczy przedmiot prowadzony w ramach jednego semestru, zostanie więc podzielony na tyle instancji wykładu w elektronicznym dzienniku ilu jest prowadzących i wykladowców. Każdy z nich będzie zbierał obecności tylko w ramach prowadzonych przez siebie zajęć.

Jeśli użytkownik elektronicznego dziennika, w ramach jednego przedmiotu prowadzi wiele różnych zajęć. np. wykład i ćwiczenia albo ćwiczenia z różnymi grupami studenów, powinien on przypisać do swojego konta kilka instancji wykładu z przypisanymmi do niej grupami studenów oraz instancjami prowadzonych przez niego zajęć z tą grupą.

Interfejs użytkownika

Interfejs użytkownika elektronicznego dziennika jest prosty i sklada się tylko z elementów umożliwiający tworzenie struktury instancji wykładów i zajęć oraz zarządzaniem obecnościami. Wygląd strony został oparty na frontendowym freamworku Bootstrap[9]. Podstrony zostały ostylowane w sposób zapewniający przejrzystość i w miarę nowoczesny wygląd.

Cały interfejs aplikacji został zrealizowany w jezyku angielskim i na chwilę obecną nie ma możliwości przełączenia sie na język polski.

Główną podstroną widoczna dla użytkownika po zalogowaniu się na konto jest przegląd instancji wykładów przypisanych do niego. Z tego poziomu mogą zostać wykonane standardowe funkcje CRUD[**?**].

Formularz dodawania nowej instancji wykładu umożliwia załączenie pliku, na postawie którego ostanie zainicjaliwowana grupa uczstników instancji wykładu. Plik powinien zawierać listę numerów albumu studentów wypisanych po przecinku.

Inicjalizacja listy zajęć jest możliwa w jednym z trzech trybów - pojedynczym, cotygodniowym lub comiesięcznym. W razie wybrania którejś z dwóch ostatnich opcji formularz umożliwi podanie początkowej i końcowej daty, a do wykładu zostaną przypisane zajęci odpowiednio co siedem i co 30 dni od pierwszej z nich. Formularz zakłada, że wszystkie zajęcia odbywać się będą o tej samej godzinie. Użytkownik będzie miał możliwość zmiany godziny w przeglądzie poszczególnych zajęć.

Po wejściu w przegląd instancji wykladu aplikacja wyświetla informacje o jej nazwie i opisie. Dodatkowo w dwóch domyślnie zwiniętych listach wyświetlona zostaje lista obecności i lista zajęć. Użytkownik ma możliwość dodawania/usuwania elementów tych list. W przypadku kiedy student posiadał już zaliczone obecności w ramach zajęć przyporządkowanych do tej instancji wykładu, a zostanie usunięty z listy uczesników, jego uczestnictwo zostanie zinterpretoweane jako udział wolnego słuchacz w zajęciach.

W tym widoku użytkownik dostaje również możliwość wygenerowania raportu z obecnościami w formacie CSV.

Ostatnim istotnym widokiem jest przegląd pojedynczych zajęć. Użytkownik otrzymuje możliwość zmodyfikowania ich daty i godziny oraz wyświetlenia i zarządzania listą obecności. Lista podzielona jest na dwie części - studentów którzy są uczestnikami instancji wykładu oraz wolnych słuchaczy. Każdy wiersz na pierwszej z list pokolorowany jest na czerwono lub zielono w zależności od tego czy dany uczesnik był obecny na zajęciach.

Dodatkowo interfejs umożliwia w tym widoku wysłanie na serwer pliku z obecnościami które zostały wygenerowane prze weryfikator.

Połączenie z bazą studentów

Jak zostało określone podczas definiowania wymagań wobec projektu, dziennik elektroniczny jest niezależny od danych dotyczących studentów i nie przechowuje ich w żadnej formie.

Podczas dodawania studenów do listy uczesników wykładu albo listy obecności na zajęciach aplikacja odwołuje się do zewnętrznego serwisu udostępniającego dane studentów. Korzystając z podanych przez użytkownika informacji identyfikujących studenta - numeru albumu albo UIDu karty magnetycznej, odpytuje zewnętrzną aplikację korzystając z RESTowego api. Kiedy dane studenta zostaną poprawnie pobrane, dziennik zapamiętuje jedynie identyfikator studenta w zewnętrznym systemie. Przy każdym żądaniu wyświetlenia danych studentów aplikacja wykorzystuje zapamiętany identyfikator do wysłania zapytania do zewnętrznego serwisu.

Takie podejście ma zasadnicze korzyści. Dzięki zapamiętywaniu zewnętrznego identyfikatora, aplikacja może uniknąć zapamiętywania wrażliwych danych. Pozwala to znacząco ograniczyć straty nawet w przypadku naruszenia bezpieczeństwa serwisu. Dodatkowo taka architektura jest przygotowaniem systemu do współpracy z rzeczywistami źródłami danych o studentach i wpięcia do cyfrowego systemu istniejącej uczelni.

Przegląd technologii

Aplikacja została napisana we frameworku Yii2[13], będącym rozszerzeniem języka PHP. Cała architektura opiera się na modelu MVC dodatkowo w ramach definiowania modeli i łączenie ich z bazą korzystam ze wzorca ActiveRecord, a kod związany z przetważaniem innych źródeł danych w serwisie wydzieliłem do osobnych serwisów w ramach aplikacji.

W projekcie frontendu korzystam z frameworka Bootstap[9], oraz gotowych komponentów[10][11][12] wyświetlających edytowalne pola z datą i godziną.

Dane wygenerowane w trakcie funkcjonowania aplikacji przechowuję w relacyjnej bazie danych operującej na systemie PostreSQL[15].

Gotowa aplikacja została uruchomiona w środowiku najnowszej wwesji PHP - 7.4.

Do zarządzania zależnościami serwisu wykorzysuję system do zarządzania pakietami Composer[14].

4.3. Baza studentów

Rozdział 5.

Możliwość dalszego rozwoju systemu

Zautomatyzowana weryfikacja

Mankamentem zaprezentowanego w powyższej pracy podejścia jest konieczność manualnej weryfikacji czy osoba zbliżająca legitymację do czytnika jest rzeczywiście jej właścicielem. W opracowaniu [**?**], przedstawiono projekt udostępniania informacji o studencie w oparciu o kartę RFID oraz dodatkową weryfikację poprzez sprawdzenie linii papilarnych. W systemie weryfikującym obecność na zajęciach można byłoby skorzystać z podobnego rozwiązania.

Innym dającym się zastosować podejściem jest system do rozpoznawania twarzy.

Integracja z uczelnianym systemem

Projektowany przeze mnie system od początku pomyślany był w taki sposób, aby umożliwić integrację z zewnętrznym źródłem informacji o studentach. Bazę studentów, która w moim projekcie jest osobną aplikację, łatwo można by zastąpić przez akademicką platformę. Kolejnym polem do integracji jest system logowania. Korzystanie z konta z USOSa, byłoby ułatwieniem dla korzystających z aplikacji pracowników i studentów oraz pozwoliłoby uniknąć przechowywania w systemie wrażliwych danych związanych z utworzonymi kontami.

Udostępnienie systemu uczestnikom wykładu

Rozwinięciem systemu byłoby na pewno udostępnienie studentom funkcjonalności zalogowania się i wglądu do ich aktualnych statystyk związanych z obecnościami. Byłoby to zadanie o tyle utrudnione, że należałoby udostępniać zainteresowanemu dane dotyczące tylko jego osoby. Można byłoby to rozwiązać poprzez integrację z

uczelnianym systemem do logowania albo udostępniając studentom konta zakładane przez administratora, który zajmowałby się potwierdzeniem tożsamości.

Informowanie zainteresowanych o obecnościach

W cytowanym we wstępie opracowaniu [1] wskazane jest, że sprawdzanie obecności ma znaczenie nie tylko jako forma przymusu, ale również jako wyraz troski uczelni o swoich studentów. Wskazane jest, że ‘wczesna interwencja‘ w przypadku opuszczania przez ucznia zajęć, może zapobiec przerwaniu przez niego studiów. Nasuwa to bardzo wiele możliwości rozwoju aplikacji weryfikującej obecności. Zaprojektowaną przeze mnie platformę można by wyposażyć w możliwość informowania studenta o tym, że limit jego nieobecności na pewnych zajęciach jest bliski wyczerpania się.

Możliwe również byłoby automatyczne informowanie organów uczelnianych o kłopotach studenta na podstawie jego absencji.

Platforma do szybkiego reagowania

Idąc tym tropem, integrując uczelniane systemy, udostępniając studentom wgląd w ich obecności i umożliwiając interakcje poprzez kanały takie jak mail czy sms, można z prostej aplikacji stworzyć rozbudowaną platformę do monitorowania wyników studenta i adekwatnego reagowania w przypadku wykrycia potencjalnych problemów. Powiązując informacje na temat tego na jakim etapie studiów znajduje się student z jego aktualnymi wynikami, byłoby możliwe automatyczne dopasowanie pomocy, takiej jak rozmowa z pracownikiem uczeni odpowiedzialnym za studentów, czy zaproponowanie dodatkowych zajęć

Rozdział 6.

Podsumowanie

1. Żaden z przytoczonych przykładów nie rozwiązuje problemu w tak kompleksowy sposób 2. Zbieranie danych na pendrive co jest bardziej elastyczne

...

Bibliografia

[1] Debbie Bevitt and Chris Baldwin and Jane Calvert, *Intervening Early: Attendance and Performance Monitoring as a Trigger for First Year Support in the Biosciences* 2016, Bioscience Education https://doi.org/10.3108/beej.15.4

[2] Vishal Bhalla and Tapodhan Singla and Ankit Gahlot and Vijay Gupta *Bluetooth Based Attendance Management System* 2013 http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.645.1573

[3] Oo, Zaw Lin and Win Lai, Theint and Than, Maung Maung *Web Server Base RFID Attendance Record Systems* 2018 https://www.researchgate.net/publication/333115623_Web_Server_Base_RFID_Attendance_Record_System/comments

[4] Patel, Unnati and Swaminarayan, Priya *Computer Science and Management Studies Development of a Student Attendance Management System Using RFID and Face Recognition: A Review* 2014, International Journal of Advance Research in Computer Science and Management Studies https://www.researchgate.net/publication/325819350_Computer_Science_and_Management_Studies_Development_of_a_Student_Attendance_Management_System_Using_RFID_and_Face_Recognition_A_Review

[5] Arulogun, Oladiran and Olatunbosun, Adeboye and A., Fakolujo and Olayemi Mikail, Olaniy *RFID-Based Students Attendance Management System* 2013, International Journal of Engineering and Scientific Research https://www.researchgate.net/publication/235598499_RFID-Based_Students_Attendance_Management_System

[6] Shoewu, Oluwagbemiga and Makanjuola, N. and Ajasa, Abiodun and Oluwafemi J., Ayangbekun *Design and Implementation of an Rfid Based Automated Students Attendance System R BASAS* 2015, JOURNAL OF ADVANCEMENT IN ENGINEERING AND TECHNOLOGY https://www.researchgate.net/publication/235598499_RFID-Based_Students_Attendance_Management_System

[7] https://en.wikipedia.org/wiki/Radio-frequency_identification

[8] ZARZĄDZENIE Nr 116/2020 Rektora Uniwersytetu Wrocławskiego z dnia 3 września 2020 r https://bip.uni.wroc.pl/download/attachment/25560/nr_116_2020_zarzadzenie_funkcjonowanie-uwr-w-zwiazku-z-covid_19.pdf

[9] https://getbootstrap.com/

[10] https://github.com/kartik-v/yii2-widget-timepicker

[11] https://github.com/2amigos/yii2-date-time-picker-widget

[12] https://github.com/2amigos/yii2-date-picker-widget

[13] https://www.yiiframework.com/

[14] https://getcomposer.org/

[15] https://www.postgresql.org/