AKADEMIA GÓRNICZO-HUTNICZA

Raport z projektu

Alarm domowy

z przedmiotu

Sensory w Aplikacjach Wbudowanych

Elektronika i Telekomunikacja - Systemy Wbudowane, rok I studiów magisterskich

Tomasz Bednorz

Jakub Rajs

Cezary Szczepański

23.06.2023

Spis Treści:

1. Opis i założenia projektu

1.1. Założenia projektowe i wysokopoziomowy opis projektu

Celem projektu jest realizacja wielofunkcyjnego niezawodnego alarmu domowego. Alarm może może mieć wiele zastosowań w zależności od jego umiejscowienia, takich jak:

- wykrycie otworzenia okna,
- wykrycie podniesienia np. skrzyni,
- przekroczenie przez człowieka lub zwierzę wiązki światła/dźwięku.

Jego niezawodność polega na realizacji pomiarów przy pomocy dwóch metod:

- czujnika ultradźwiękowego (pomiar odległości),
- fotokomórki IR (odczyt wartości 1/0).

Kolejnym elementem czyniącym system niezawodny miała być komunikacja przy pomocy dwóch niezależnych protokołów komunikacyjnych. Niestety w ostatecznej implementacji został zastosowany tylko CAN (Bluetooth nie został zrealizowany z powodu trudności implementacyjnych).

Akwizycja danych z magistrali oraz ich prezentacja została zrealizowana przy użyciu biblioteki Tkinter w Pythonie. Na tę część składały się:

- odbieranie danych z konwertera CAN2USB poprzez port szeregowy
- komunikacja z bluetooth
- przetwarzanie surowych danych do formatu gotowego do zaprezentowania
- stworzenie prostego interfejsu graficznego

Ze względu na zastosowanie jedynie CANa oraz UARTa interfejs ostatecznie nie obsługuje Bluetootha.

Założenia:

- niezawodność pomiarowa: czujnik ultradźwiękowy oraz fotokomórka IR,
- niezawodność komunikacyjna: CAN oraz Bluetooth (nie zrealizowane),
- filtrowanie danych na poziomie platformy wbudowanej,
- interpretacja danych na poziomie GUI

1.2. Podział odpowiedzialności

Tabela 1. Podział funkcjonalności dla poszczególnych członków projektu

Osoba	**Funkcjonalności**
Tomasz Bednorz	Oprogramowanie na platformę wbudowaną ESP32. Pobór danych z czujnika ultradźwiękowego HY-SRF05 oraz fotokomórki. Transmisja danych z użyciem magistrali CAN.
Jakub Rajs	Akwizycja danych na stację roboczą przez CAN oraz UART oraz GUI.
Cezary Szczepański	Dokumentacja oraz praca nad GUI
Stanisław Zachorowski	Hardware

2. Sprzęt

2.1. Wybór komponentów

Mikrokontroler wybrany do realizacji projektu to ESP32 (rysunek 1). Został wybrany z uwagi na niską cenę (20zł) oraz łatwość montażu. Początkowo realizowana miała być komunikacja przy pomocy Bluetooth, a wybrany układ nie wymaga dodawania na płytce drukowanej transceivera radiowego oraz t2oru radiowego.. Wybrany układ ESP32 posiada wszystkie wymagane peryferia do realizacji projektu (CAN, timery).

Rys 1. Układ WiFi + Bluetooth BLE Espressif ESP32-WROOM-32E, Botland [1]

Kolejnym wybranym komponentem do realizacji projektu jest czujnik ultradźwiękowy HY-SRF05 (rysunek 2). Przy jego pomocy mierzona jest odległość do najbliższej przeszkody i na tej podstawie interpretowane jest przecięcie wiązki sygnału.. Umożliwia pomiar w zakresie 2-400 cm, co jest wystarczającą odległością w realizowanym projekcie. Pomiar dokonywany jest poprzez wystawienie stanu wysokiego na pin TRIG na 10us, a następnie pomiar czasu wystawienia przez czujnik stanu wysokiego na pin ECHO. Przy pomocy wzoru na prędkość dźwięku można obliczyć dystans od przeszkody.

$$\text{Distance [cm]} = \text{ECHO}_{\text{HIGH}}\ \text{[us]} * 0.0343\ \text{[cm/us]} / 2$$

Np. Dla stanu wysokiego wystawionego przez 3.5ms odległość wyniesie:

$$\text{Distance} = 3500\ \text{[ms]} * 0.0343\ \text{[cm/us]} / 2 = 60.025\ \text{[cm]}$$

Pomiar czasu został zrealizowany przy pomocy sprzętowego timera. Start/stop pomiaru czasu był wyzwalany poprzez zbocze narastające/opadające pinu ECHO.

Rys 2. Ultradźwiękowy czujnik HY-SRF05, Botland [2]

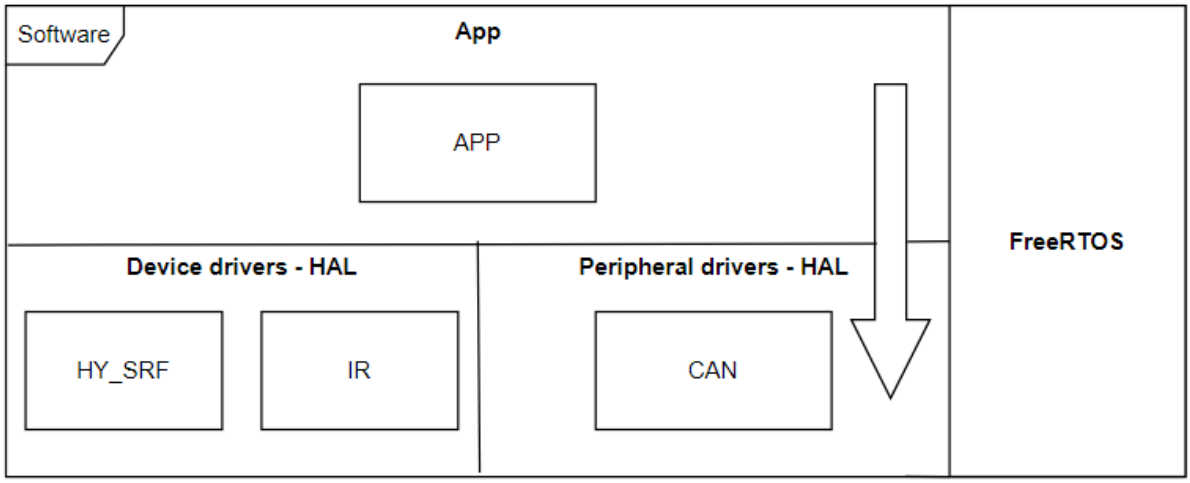
Ostatnim wykorzystywanym komponentem jest czujnik przerwania wiązki IR (rysunek 3). Jego działanie jest bardzo proste. W momencie przerwania wiązki światła podczerwonego pomiędzy nadajnikiem, a odbiornikiem następuje zmiana stanu logicznego wyjścia OUT.

Rys 3. Fotokomórka do bram - czujnik przerwania wiązki IR, Botland [3]

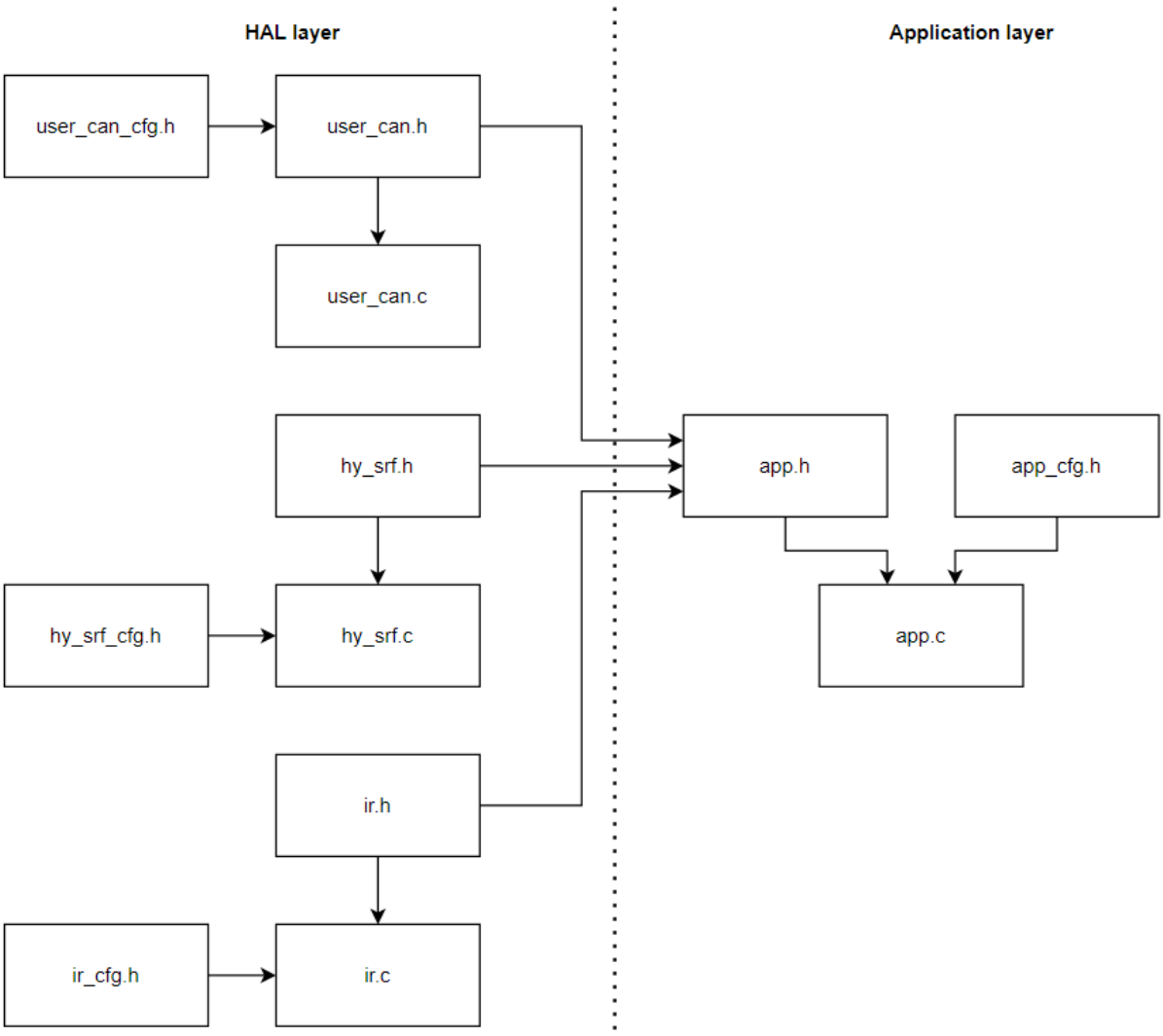
Wybrane elementy zostały wybrane z uwagi na niską cenę oraz funkcjonalność wystarczającą do realizacji projektu.

3. Oprogramowanie

3.1. Oprogramowanie wbudowane



Rys 4. Architektura oprogramowania - przepływ danych



Rys 5. Architektura oprogramowania - struktura plików

Oprogramowanie na platformę wbudowaną składa składa się z czterech głównych modułów:

- CAN - warstwa abstrakcji na peryferium komunikacyjne CAN,
- HY_SRF - warstwa abstrakcji dla czujnika ultradźwiękowego,
- IR - warstwa abstrakcji dla czujnika IR,
- APP - warstwa aplikacji.

Moduły CAN, HY_SRF i IR wykorzystują gotowe biblioteki sterowników ESP32 do obsługi: gpio, can-a oraz timerów. Dodatkowo zawierają moduły konfiguracyjne umożliwiające szybką zmianę np. podpiętych pinów, okresu wywoływania tasków, lub ilości próbek wchodzących do kolejki uśredniającej/debouncingowej.

Tabela 2. Pobór danych z czujników

Czujnik	**Częstotliwość akwizycji danych**	**Rodzaj filtracji**	**Ilość próbek**
Ultradźwiękowy (HY-SRF05)	40 Hz	Uśrednianie	3
IR	200 Hz	Debouncing	5

Tabela 2 prezentuje częstotliwość akwizycji danych oraz rodzaj filtracji dla poszczególnych czujników. Niska częstotliwość poboru danych z czujnika ultradźwiękowego wynika z długiego czasu pojedynczego pomiaru odległości. Dla maksymalnego zasięgu (400 cm) wynosi ponad 23 ms. Zastosowano dwa różne rodzaje filtracji dla poszczególnych czujników:

- uśrednianie - obliczanie średniej z 3 kolejnych próbek. Najnowszy pomiar wchodzi na początek kolejki uśredniającej, a najstarszy przestaje być brany pod uwagę podczas obliczania średniej,
- debouncing - zmiana stanu gdy 5 kolejnych próbek jest innych od stanu aktualnie zdebouncingowanego.

Przefiltrowane dane są transmitowane przy pomocy magistrali CAN. Transmisja jest niezależna od tego czy pojawiły się nowe dane lub zmieniała się wartość po ich przefiltrowaniu. Na magistralę dane transmitowane są ciągle - zawsze najnowsza, przefiltrowana wartość. Baudrate transmisji został ustawiony na sztywno 250 Kbit.
Tabela 3 przedstawia ramki występujące na magistrali CAN. Dwie ramki wysyłane są naprzemiennie co 5 ms. Ramka z danymi z czujnika ultradźwiękowego zawiera odległość w cm, dlatego DLC=2 (maksymalna zmierzona odległość wynosi ponad 255 cm). Ramka z danymi z czujnika IR zawiera wartość 1/0, więc DLC=1 jest wystarczające.

Tabela 3. Ramki CAN

Nazwa ramki	ID	Okres [ms]	Offset [ms]	DLC (długość w bajtach)
UserCanFrame1_HySrf	0x05	10	0	2
UserCanFrame2_Ir	0x10	10	5	1

Poprzez zastosowanie X-MACRO'ów możliwe jest łatwe i szybkie konfigurowanie ramek wysyłanych magistralą CAN.

Poniższy fragment kodu przedstawia define konfigurujący ramki. Umożliwia on łatwe dodawanie kolejnych z łatwą konfigurowalnością parametrów.

```
/* CAN frames configuration: name, ID, period (ms), offset (ms), dlc */
#define USER_CAN_CFG_TABLE  \
    USER_CAN_CFG_FRAME(UserCanFrame1_HySrf, 0x05, CAN_USER_TIME_TO_MS(10U), 
CAN_USER_TIME_TO_MS(0U), 2U)   \
    USER_CAN_CFG_FRAME(UserCanFrame2_Ir, 0x10, CAN_USER_TIME_TO_MS(10U), 
CAN_USER_TIME_TO_MS(5U), 1U)
```

Poniższe dwa fragmenty kodu pokazują jak łatwo przy pomocy X-Macr można stworzyć enuma zawierającego zdefiniowane ramki oraz tablicę zawierającą struktury konfiguracyjne dla poszczególnych ramek. Dodanie kolejnych ramek wymaga modyfikacji kodu tylko w jednym miejscu, a konkretnie define'a USER_CAN_CFG_TABLE. X-Macra mają tą samą nazwę, gdyż define USER_CAN_CFG_TABLE wykorzystuje nazwę USER_CAN_CFG_FRAME podczas tworzenia enuma oraz tablicy struktur konfiguracyjnych.

```
typedef enum
{
    #define USER_CAN_CFG_FRAME(name, id, period, offset, dlc)   name,
        USER_CAN_CFG_TABLE
    #undef USER_CAN_CFG_FRAME
    UserCanFrameMax
}UserCan_Frame_t;
```

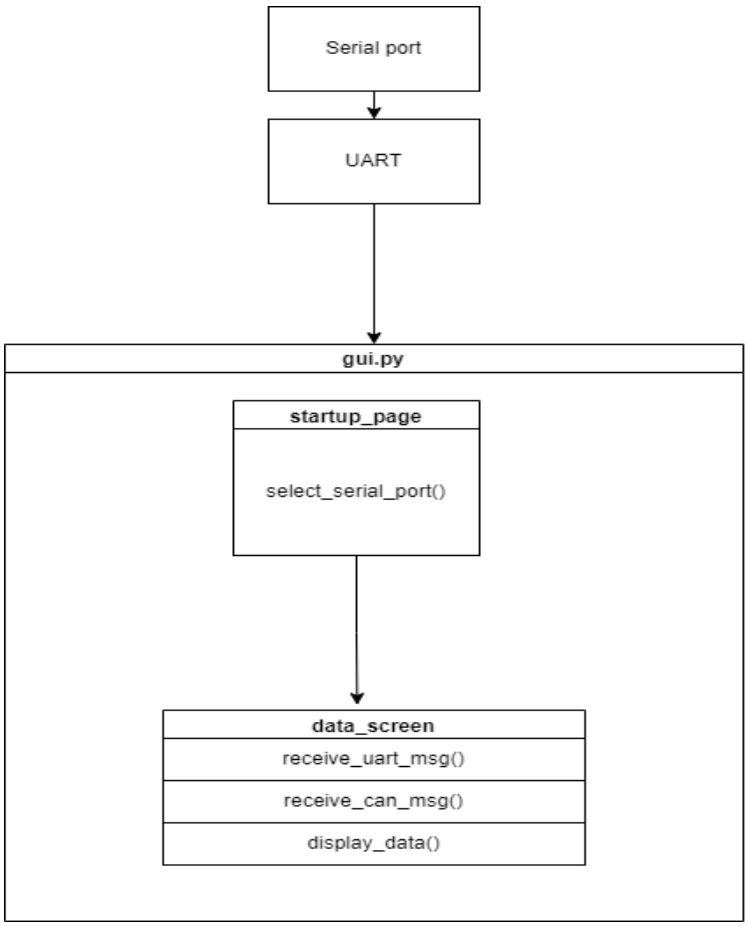
```
UserCan_Config_t Can_UserConfig[UserCanFrameMax] =
{
    #define USER_CAN_CFG_FRAME(name, id, time, offset, length) { \
        .frame_name=name, \
        .message={.identifier=id, .data_length_code=length, .data={0}}, \
        .period=time, \
        .counter=offset},
        USER_CAN_CFG_TABLE
    #undef USER_CAN_CFG_FRAME
};
```

Stworzenie interfejsu uzupełniającego ramki danymi staje się w tym wypadku bardzo proste.

```
void UserCan_FillFrame(UserCan_Frame_t frame_id, uint8_t* data);
```

3.2. GUI

W pierwszej kolejności, aby otworzyć port szeregowy należy wybrać odpowiedni numer z dostępnej listy. Program automatycznie wykrywa dostępne porty i daje użytkownikowi możliwość wyboru. Po kliknięciu w odpowiedni przycisk następuje przejście do ekranu, na którym wyświetlane są dane.



Rys.6. Schemat akwizycji i prezentacji danych

Dane są przesyłane przez port szeregowy przy baudrate 115200. Samo przetwarzanie jest realizowane w pętli while, której prędkość zależy głównie od urządzenia, na którym uruchomiono program. Pomiary wykonane na stacji roboczej wykazały średnią częstotliwość rzędu 10^6, co jest wartością wystarczającą na potrzeby wyświetlania danych.

Dzięki wykorzystaniu odpowiednich bibliotek - *serial* oraz *python-can* nie trzeba było samodzielnie implementować odczytu ramek danych. Zamiast tego przy użyciu tych bibliotek można było zapisywać odczytane dane bezpośrednio do zmiennych w kodzie.

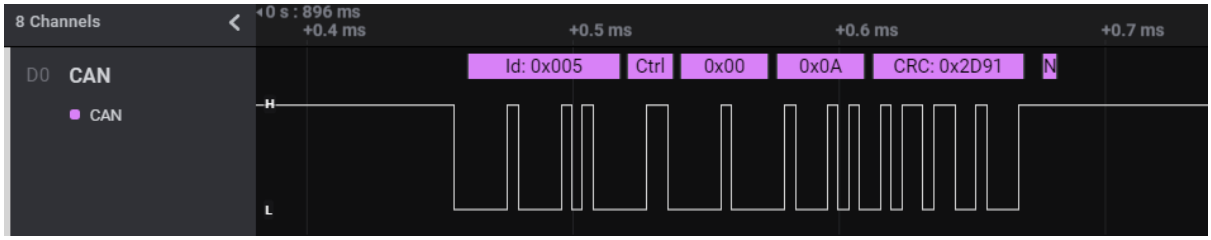
W zależności od ID ramki dane były wpisywane w odpowiedni widget w GUI. Jeśli ID nie zgadzało się z uzgodnionymi wcześniej wartościami, to dane nie były zapisywane. Format danych sprawił, że uznano, że najwygodniej będzie odczytać dane linia po linii. W ten sposób uzyskano paczki danych składające się z ID oraz danych rozdzielone spacją. Po rozdzieleniu i odcięciu zbędnych znaków nowej linii uzyskano odpowiednie dane.
Tak zebrane i przetworzone dane przekazano do metod obsługujących interfejs graficzny.

Przy każdym wykryciu odpowiedniego ID aktualizowana jest wartość wpisana w widget oraz, jeśli dane są nieprawidłowe lub wykryta odległość jest mniejsza niż 50cm, to wyświetlone jest czerwone koło (“lampka”), natomiast gdy odległość jest większa, to koło jest zamieniane na zielone.

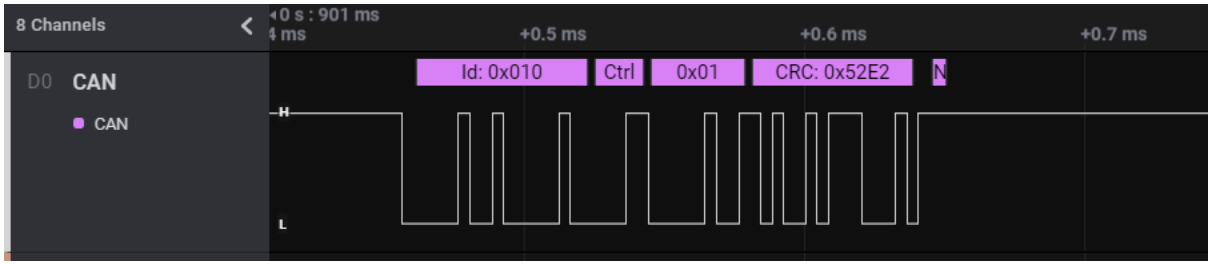
4. Uruchomienie

4.1. Oprogramowanie wbudowane

Podczas realizacji projektu zostały przeprowadzone testy systemu z użyciem płytki rozwojowej ESP32-DevKitC zawierającej moduł ESP32-WROOM. Poniżej znajdują się zrzuty ekranu z analizatora stanów logicznych przedstawiających komunikację na magistrali CAN.



Rys 7.Ramka UserCanFrame1_HySrf, 0x0A = 10cm



Rys 8.Ramka UserCanFrame2_Ir, 0x01 = ON

Rys 9.Ramki w odstępie co 5ms

Podczas realizacji zaimplementowanych funkcjonalności nie napotkano większych problemów. Początkowo planowano transmisję danych dodatkowo przy użyciu Bluetooth. Niestety kilkugodzinne starania próby przetestowania przykładowego oprogramowania z gotową implementacją zakończyły się niepowodzeniem (trudności z odbieraniem danych przy pomocy oprogramowania testującego m. in. nRF Connect).

4.2. GUI

Rys.10. Ekran wyboru portu szeregowego

Rys.11. Ekran z wyświetlaniem danych z wykrytą przeszkodą

Rys.11. Ekran z wyświetlaniem danych bez wykrytej przeszkody

Przy implementacji napotkano problem z odczytem danych z wykorzystaniem konwertera CAN2USB[5]. Testy z wykorzystaniem analizatora stanów logicznych wykazały, że możliwa jest transmisja danych ze stacji roboczej, jednak odczyt nie działał na poziomie konwertera. Z tego powodu pomimo teoretycznie sprawnej implementacji metody do odczytu danych przez CAN nie została ona przetestowana w praktyce. Niemniej jednak odczyt danych bezpośrednio przez UART został przetestowany i działa poprawnie.

Bibliografia

1. Botland, “Układ WiFi + Bluetooth BLE Espressif ESP32-WROOM-32E - SMD - 32 Mbit - 4 MB Flash”, dostęp: 12.06.2023r.

2. Botland, “Ultradźwiękowy czujnik odległości 2-400cm - HY-SRF05”, dostęp: 12.06.2023r.

3. Botland, “Fotokomórka do bram - czujniki przerwania wiązki IR”, dostęp: 12.06.2023r.

4. Elty, “USB To CAN Adapter Model - Adapter USB do magistrali CAN bez obudowy”,, dostęp: 12.06.2023r.

5. Repozytorium kodu, “Home_alarm”, dostęp: 23.06.2023r.