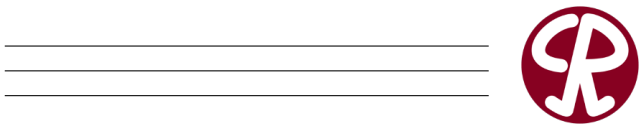
Projekt

Wizualizacja Danych Sensorycznych

Płytka odbijająca piłkę

Marcel Konieczny, 252966



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i Mikrosystemów
Politechniki Wrocławskiej

27 kwietnia 2022

Spis treści

1 Charakterystyka tematu projektu **1**

2 Wykonane prace wstępne **1**

3 Postęp prac **6**

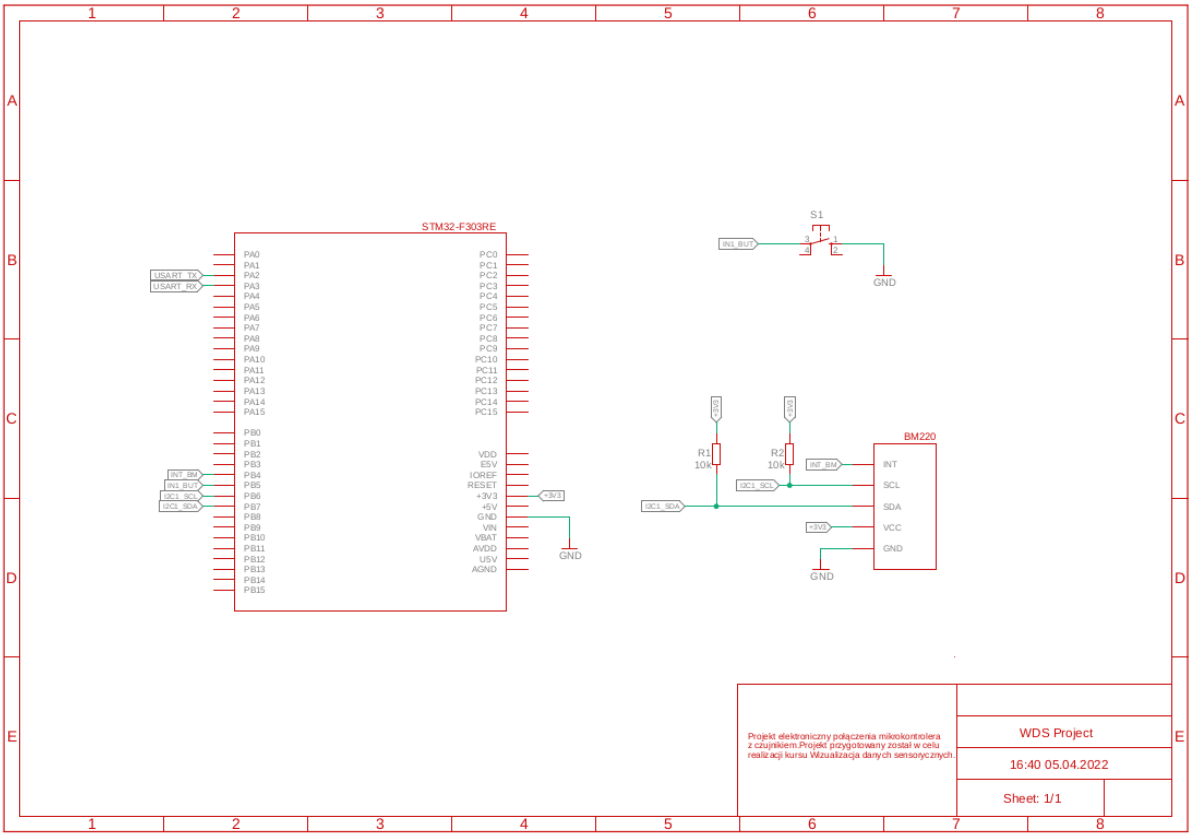
1 Charakterystyka tematu projektu

Celem projektu jest wizualizacja płytki odbijającej piłkę, płytka będzie się poruszała w 4 kierunkach. Płytka będzie sterowana na podstawie odczytów z akcelerometru podłączonego do mikrokontrolera STM32. Aby poruszyć płytką w aplikacji w prawą lub lewą stronę będzie konieczne trzymanie przycisku i poruszenie akcelerometrem w odpowiednią stronę. Poruszenie akcelerometrem w dół będzie powodowało, że podczas odbicia kula będzie zwolniona, poruszenie akcelerometrem w górę spowoduje przyśpieszenie kuli. Zwolnienie i przyśpieszenie kuli będzie odbywać się bez wciśniętego przycisku. Płytka którą użytkownik będzie mógł sterować będzie znajdować się na spodzie interfejscu, na górze interfejsu będzie znajdować się płytka sterowana przez komputer (będzie sterowana na podstawie algorytmu).

2 Wykonane prace wstępne

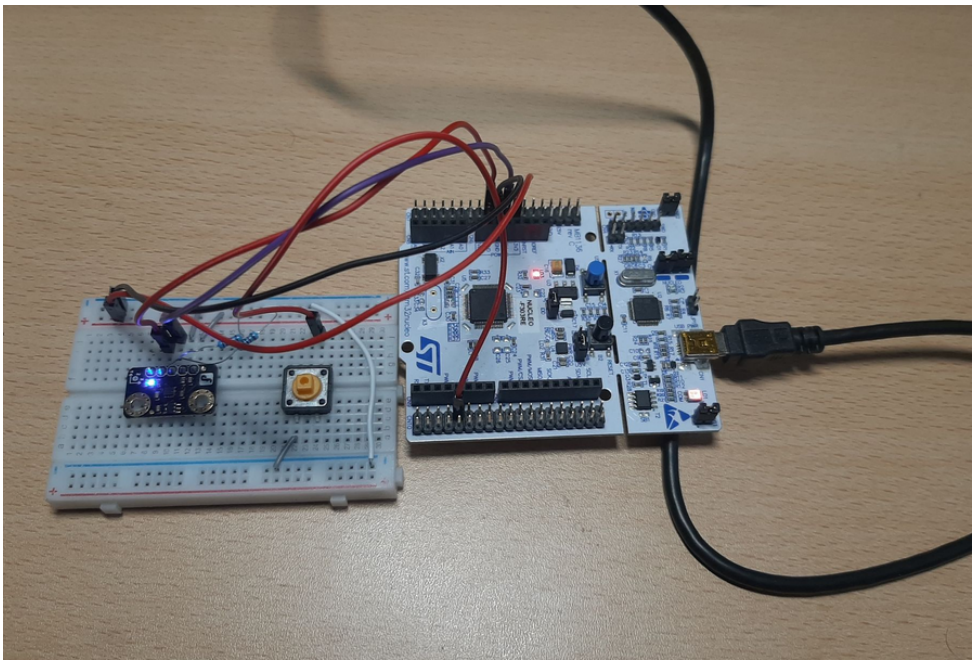
- Wykonano projekt graficzny aplikacji
  Projekt graficzny aplikacji został przedstawiony w poprzednim kamieniu milowym.

- Schemat układu elektronicznego
  W celu realizacji projektu wykonano następujący schemat elektroniczny



Rysunek 1: Schemat elektroniczny

- Utworzenie prototypu układu na płytce stykowej
  W celu realizacji etapu wykonano układ elektroniczny na płytce stykowej



Rysunek 2: Układ elektroniczny na płytce stykowej

- Oprogramowanie kompunikacji komputer-STM32 za pomocą UART.
  W celu realizacji etapu należało napisać bibliotekę do obsługi akcelerometru za pomocą której mikrokontroler STM32 jest w stanie odczytywać dane na temat położenia czujnika.

  Następnie należało zaimplementować wysyłanie danych za pomocą peryferium UART. Zdecydowałem że przesyłane dane będą w formacie:

$$"X \quad (p.o.\ x) \quad (p.o.\ y) \quad (p.o.\ z) \quad CRC8"$$

  Znaczenie poszczególnych elementów przesyłanych danych:

  – X - Znak początkowy sygnalizujący że następuje przesył danych,
  – p.o - Przyspieszenie względem osi,
  – CRC8 - Suma kontrolna zapisywana w systemie szesnastkowym,

  Suma kontrolna CRC8 jest obliczana na podstawie danych. Kod wykorzystany w celu wysyłu oraz wygenerowania kodu CRC8:

```
uint16_t CRCSingleByte(uint16_t data) {
  uint16_t poly = (POLYNOMIAL << 7);
  for (uint8_t i = 0; i < 8; i++) {
    if ((data & 0x8000) != 0)
      data ^= poly;
    data <<= 1;
  }
  return data;
```

```
}

uint16_t CalculateCRC8(uint8_t *ptr) {
  uint16_t data = ptr[0] << 8;

  for (int i = 1; i < strlen(ptr); i++) {
    data |= ptr[i];
    data = CRCSingleByte(data);
  }
  data = CRCSingleByte(data);
  return (data >> 8);
}

void printValues(BMA220 acc) {
  uint8_t *FChar = "X "; //First char
  uint8_t *EChar = " \n"; //End chars
  uint16_t CRC8 = 0;
  static uint8_t buf[50];
  static uint8_t help[25];

  strcpy(buf, FChar); //Copy "X " to bufor
  sprintf(help,"%d %d %d",acc.accData[0],acc.accData[1],acc.
     accData[2]);
  strcat(buf, help);  //Copy message to bufor
  CRC8 = CalculateCRC8(buf);
  sprintf(help, " %X", CRC8);
  strcat(buf, help);
  strcat(buf, EChar); //Copy end chars to buf

  HAL_UART_Transmit(&huart2, buf, strlen(buf), 1000);
}
```

Aby odczytywać dane na komputerze stworzono za pomocą biblioteki qt tymczasowe okno głowne, zawierające widget z tekstem oraz przyciski: rozpoczynający transmisje danych i zamykający okno główne.

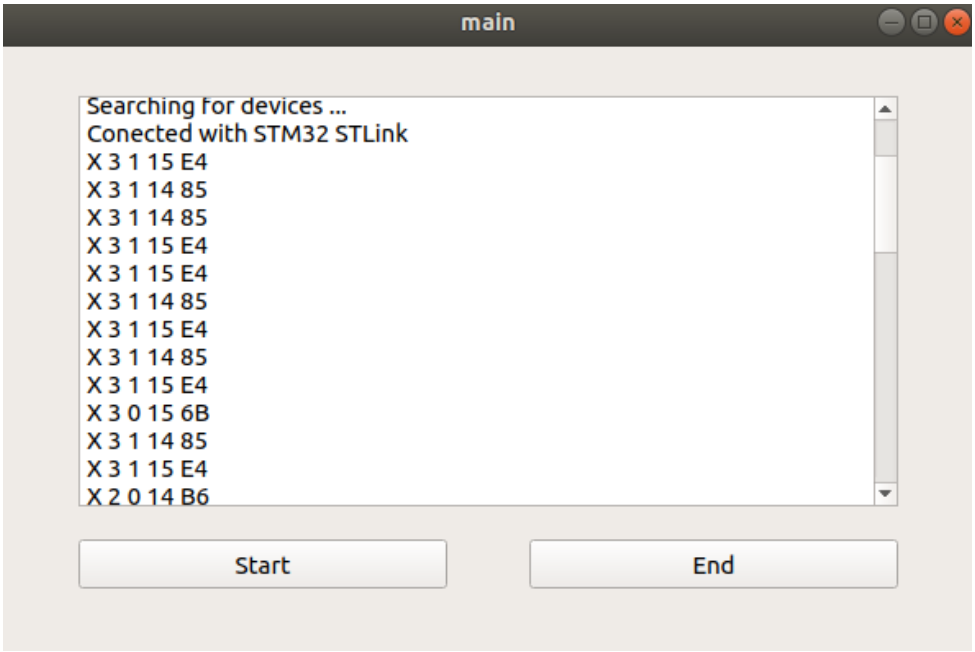
```
    //Okno z widgetem
UARTWindow::UARTWindow(QWidget* parent) {
    this->CentralWidget = new UARTRead(this);
    this->device = new QSerialPort(this);
    setCentralWidget(CentralWidget);
    resize(600, 400);
    connect(CentralWidget, SIGNAL(EmitClosing()), this, SLOT(
       close()));
    connect(CentralWidget, SIGNAL(EmitStartUART()), this, SLOT(
       ConnectDevice()));
}


//Widget z przyciskiem i polem tekstu
UARTRead::UARTRead(QWidget* parent) :QWidget(parent)
{
```

```
15      QPushButton* buttonStart = new QPushButton(tr("Start"), this)
            ;
16      QPushButton* buttonEnd = new QPushButton(tr("End"), this);
17      this->text = new QTextEdit(tr("Dane z czujnika"), this);
18
19      this->text->setGeometry(50, 30, 500, 250);
20      buttonEnd->setGeometry(325, 300, 225, 30);
21      buttonStart->setGeometry(50, 300, 225, 30);
22      this->text->setReadOnly(true);
23      connect(buttonEnd, SIGNAL(clicked()), this, SLOT(EndProgram()
            ));
24      connect(buttonStart, SIGNAL(clicked()), this, SLOT(
            StartTransmit()));
25  }
```

W celu odbioru danych użyto elementów QTSerial Port

```
1  void UARTWindow::ConnectDevice() {
2      if (device->isOpen()) {
3          CentralWidget->EditText("Closing conection");
4          device->close();
5
6          return;
7      }
8      CentralWidget->EditText("Searching for devices ...");
9      QList<QSerialPortInfo> devices;
10     devices = QSerialPortInfo::availablePorts();
11     if (devices.count() == 0) {
12         CentralWidget->EditText("No devices to connect");
13         return;
14     }
15     int devNum = 0;
16     for (devNum = 0;devNum < devices.count();devNum++) {
17         qDebug() << devices.at(devNum).portName() << devices.at(
               devNum).description();
18         if (devices.at(devNum).description() == "STM32 STLink")
19             break;
20     }
21     QString portName = devices.at(devNum).portName();
22
23     device->setPortName(portName);
24
25     if (device->open(QSerialPort::ReadWrite)) {
26         device->setBaudRate(QSerialPort::Baud115200);
27         device->setDataBits(QSerialPort::Data8);
28         device->setParity(QSerialPort::NoParity);
29         device->setStopBits(QSerialPort::OneStop);
30         device->setFlowControl(QSerialPort::NoFlowControl);
31         CentralWidget->EditText("Conected with " + devices.at(
               devNum).description());
32         connect(this->device, SIGNAL(readyRead()), this, SLOT(
               ReadTransmision()));
```

```
33      }
34      else {
35          CentralWidget->EditText("Can't connect with device");
36      }
37  }
38
39  void UARTWindow::ReadTransmision() {
40      uint16_t CRC8;
41      std::string str;
42      QString hexCRC8;
43      QString terminator = " ";
44      QString lineWOHex;          //line without hex code at end
45      while (this->device->canReadLine()) {
46          QString line = this->device->readLine();
47          if (line.at(0) != "X")
48              continue;
49          int pos1 = line.lastIndexOf(terminator);
50          lineWOHex = line.left(pos1);
51          int pos2 = lineWOHex.lastIndexOf(terminator);
52          lineWOHex = line.left(pos2);
53          str = line.left(pos2).toStdString();
54          CRC8 = CalculateCRC8(const_cast<char*>(str.c_str()));
55          hexCRC8 = QString::number(CRC8, 16).toUpper(); //Cast
                uint16_t to Qstring
56          if (hexCRC8 == line.mid(pos2 + 1, pos1 - 1).trimmed())
57              this->CentralWidget->EditText(lineWOHex);
58      }
```



Rysunek 3: Chwilowy interfejs aplikacji

3 Postęp prac

Wszystkie prace realizowane są zgodnie z założonym wcześniej planem, nie występują opóźnienia.