УНИВЕРЗИТЕТ
У НОВОМ САДУ

ФАКУЛТЕТ
ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Југославија
Деканат: 021 350-413; 021 450-810; Централа: 021 350-122
Рачуноводство: 021 58-220; Студентска служба: 021 350-763
Телефакс: 021 58-133; e-mail: ftndean@uns.ns.ac.yu

PROJEKAT

iz Računarske elektronike

TEMA PROJEKTA:

Nadzorna kamera

TEKST PROJEKTA:

Korišćenjem RPi-a, veb kamere, PIR senzora i SG90 servo motora, napraviti sistem koristeći Qt koji prenosi uživo prikaz sa kamere.

Mentor:
Prof. Ivan Mezei

Student:
David Janković, EE157-2019

U Novom Sadu, 26.8.2024.

1. Uvod

Ideja iza projekta je da se korisniku na što jednostavniji način omogući upotreba nadzornog sistema realizovanog pomoću 3 glavne komponente: servo motora, pir senzora i kamere.

Zamisao je da se putem grafičkog interfejsa kreiranog u Qt kreatoru korisniku omogući uživo prikaz sa kamere, kao i mogućnosti pomeranja ugla prikaza pritiskanjem tastera koji su u sklopu interfejsa. Takođe, korisnik ima izbor između dva režima rada, **manuelni** gde on pokreće sam nadzorni sistem i **automatski** gde senzor pokreta detektuje kretanje, koje izaziva pokretanje nadzora, koje će trajati sve dok senzor detektuje pokret. Ovakav pristup predstavlja dosta energetski efikasniji način nadzora, u slučaju kada se nadgleda neka oblast manjeg prometa.

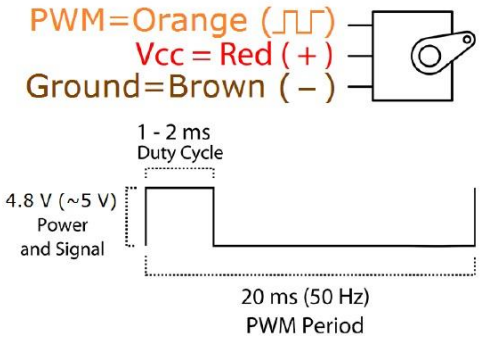
2. Komponente i alati

U narednom delu teksta dat je opis svih ključnih komponenti koje su korišćene za realizaciju porojekta.

SG90 servo motor je mikro servo motor pogodan za male projekte. Radi na naponu od 4.8V do 6V i ima mogućnosti pomeranja od oko 180 stepeni (+-90), koja se postiže dovođenjem PWM signala adekvatne dužine.

Za povezivanje motora koriste se 3 žice, predstavljene u dijagramu na *slici 1*. Kako bi se motor pokretao, potrebno je dovesti PWM signal, a trajanje pwm signala određuje ugao na koji će se motor pomeriti, kako na prijem za impuls dužine 1ms motor se obično pomeri na 0 stepeni, za 1.5ms se pomeri na 90 stepeni a za impuls dužine 2ms motor se pomera u položaj od 180 stepeni.

Kako bismo dobili PWM u kodu smo koristili SoftPMW klasu iz biblioteke WiringPi.



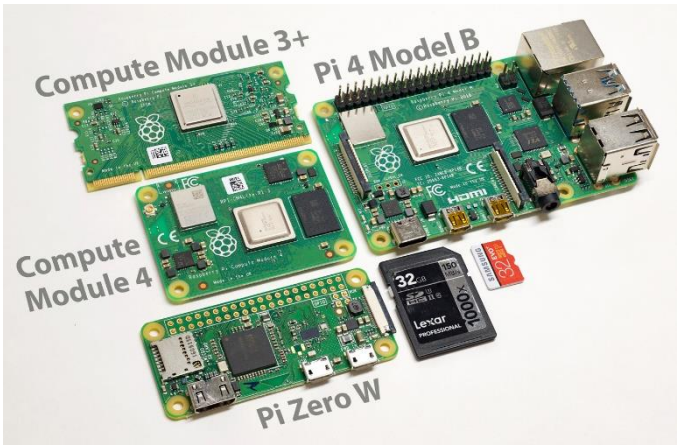
Slika 1 – Opis rada SG90 servo motora

PIR senzor (*Passive Infrared Sensor)* je digitalni elekronski senzor koji emituje infracrveno zračenje, čime je u stanju da detektuje pojavu pokreta u njegovom mernom polju. Senzor, u zavisnosti od podešavanja 2 potenciometra koja se mogu videti na slici (Slika 1), može da detektuje pokret u radiusu od 70 do 110 stepeni na razdaljini od 5 do 10 metara. Senzor radi na naponu od 5V do 3.3V, a za povezivanje na Raspberry Pi su nam neophodna 3 pina: gnd, 5v i signalni pin. U slučaju da je senzor detektovao pokret, na signalni pin se šalje signal visoke impedanse.

Slika 2 – PIR senzor

Veb kamera koju koristimo u ovom projektu je klasična PC web kamera, sa sočivom od 0.3 megapiksela i mogućnošću snimanja u 480 piksela. Kako bismo koristili njene mogućnosti u Qt kreatoru, neophodno je bilo implementirati **Qt Multimedia Module** unutar koje se nalazi **QCamera** klasa koja nam je omogućila rad sa kamerom. U narednom poglavlju će biti više reči o najbitinijim funkcijama koje su korišćene kako bi se dobio *camera-feed* za naš nadzorni sistem.

Raspberry Pi (RPI) je jednočipni računar (*Single Board Computer*) razvijen u edukativne svrhe u Velikoj Britaniji od strane Raspberry Pi fondacije. Zbog svoje pristupačnosti i svojih mogućnosti, postao je ubrzo popularan među hobistima, entuzijastima i inženjerima. Postoje različite generacije i verzije RPI-a i svaka ima svoje specifikacije i interfejse. Glavne stavke kod interfejsa skoro svih RPI-eva jesu 40 GPIO pinova, USB portovi, HDMI out portovi, Audio I/O, Eternet, dok novije generacije imaju i WiFi i Bluetooth module integrisane na samoj ploči, kao i novije verzije USB i HDMI portova. U zavisnosti od verzije, RPI na sebi može posedovati i specijalne konektore za kamere ili druge periferije.



Slika 3 - Raspberry Pi

U ovom projektu koristimo RPI 4 sa 2GB RAM-a, 2 usb 3.0 konektora, i četvorojezgarnim CPU-om od 1,56Hz.

Qt predstavlja framework za razvoj softvera koji omogućava korisnicima da kreiraju grafički korisnički interfejs (GUI) kao i aplikacije koje se mogu pokretati na više platformi. Ovaj framework je postao popularan baš zbog činjenice da je moguće kreiranje aplikacija uz minimalne promene koda za sve popularne operativne sisteme poput Windows-a, IOS-a, Linux-a i androida, pritom je open source ali se može raditi pod komercijalnom licencom.

Rad u Qz-u odlikuje Widget baziran GUI, gde se koriste widgeti kao što su dugmad, prozori, meniji i mnogi drugi UI elemnti, koji međusobno komuniciraju na bazi signal slot mehanizma. Sam Qt je razvijen u C++ programskom jeziku međutim podržava i druge jezike kao na primer Python, Java i C#, kao i razne biblioteke za njih.

3. Analiza koda

U ovom poglavlju priložen je kod pomoću kog je realizovan projekat, kao i opis ključnih delova i klasa koje su upotrebljene.

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>
#include <QtCore>
#include <QtGui>
#include <QtWidgets>
#include<QtMultimedia>
#include<QtMultimediaWidgets>
#include<wiringPi.h>
#include<QTimer>
#include<softPwm.h>

#define PIR 26
#define SG90 16


QT_BEGIN_NAMESPACE
namespace Ui { class Dialog; }
QT_END_NAMESPACE

class Dialog : public QDialog
{
    Q_OBJECT

public:
    Dialog(QWidget *parent = nullptr);
    ~Dialog();

private slots:
    void on_pushButton_Start_clicked();

    void on_pushButton_Stop_clicked();

    void on_radioButton_2_clicked();

    void on_radioButton_clicked();



    void on_motor_levo_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_0_clicked();

private:
    Ui::Dialog *ui;

    QScopedPointer<QCamera>M_Camera;
    QScopedPointer<QCameraViewfinder>M_Viewfinder;

    void start_Camera();
    void stop_Camera();
    void motion_detected();
    void moveSG90(int pwmValue);
};
#endif // DIALOG_H
```

Slika 4 – header file

Na slici 4 prikakazan je header fajl, na čijem početku su uključene u projekat sve neophodne biblioteke kao što su **wiringPi.h** pomoću kojeg dobijamo pristup GPIO portovim RPI-a **, softPWM.h** za kreiranje i kontorlu PWM signala, **QMultimedia** za rad sa kamerom i multimedia widgetima, kao i **QTimer** za kreiranje tajmera. Nakon toga definišu se pinovi za SG90 i PIR senzor. U nastavku, nalaze se svi slotovi vidžeta sa GUI-a (detaljnije o izgledu gui-a u sledećem poglavlju).

Nakon toga koristi se klasa **QscopedPointer** za upravljnje dinamički alociranim objektima **QCamera** i **QCameraViewfinder**. Objekat QCamera predstavlja kameru a **M_Camera** je pokazivač na objekat QCamera. Isto je i sa **M_ViewFinder** pokazivačem, koji predstavlja ”camera feed”.

Nakon toga, deklarisane su funkcije za pokretanje i zaustavljanje kamere, funkcija za rad sa PIR senzorom i funkcija za upravljanje servo motorom.

```
#include "dialog.h"
#include "ui_dialog.h"

int currentPosition=4;

Dialog::Dialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::Dialog)
{
    ui->setupUi(this);

    wiringPiSetup();
    softPwmCreate(SG90,0,200);


    for (const QCameraInfo &info : QCameraInfo::availableCameras())
    {
        qDebug() << "Available camera: " << info.description();
    }

    // Inicijalizacija kamere
    M_Camera.reset(new QCamera(QCameraInfo::defaultCamera()));
    qDebug() << "Default camera: " << QCameraInfo::defaultCamera().description();

    // Kreiranje viewfinder-a
    M_Viewfinder.reset(new QCameraViewfinder(this));

    // Dodavanje viewfinder-a u layout
    ui->verticalLayout->addWidget(M_Viewfinder.data());

    // Povezivanje kamere sa viewfinder-om
    M_Camera->setViewfinder(M_Viewfinder.data());

    // Postavljanje video izlaza
    QCameraViewfinderSettings viewfinderSettings;
    viewfinderSettings.setResolution(640, 480);
    viewfinderSettings.setMinimumFrameRate(15.0);
    viewfinderSettings.setMaximumFrameRate(30.0);
    M_Camera->setViewfinderSettings(viewfinderSettings);

    // Proveri podržane formate
    qDebug() << "Supported viewfinder resolutions:";
    for (const QSize &resolution : M_Camera->supportedViewfinderResolutions())
    {
        qDebug() << resolution;
    }

    // Inicijalizacija QTimer-a
        QTimer *timer = new QTimer(this);
        connect(timer, &QTimer::timeout, this, &Dialog::motion_detected);
        timer->start(1000); // Proverava svake 100 ms



}

```

Slika 5- source code prvi deo

Na slici 5 je prikazan prvi deo source code fajla, na čijem početku uključujemo prethodno kreiranu header datoteku. Nakon toga sledi inicijalizacija WiringPi biblioteke upotrebom funkcije **wiringPiSetup()** i kreiranje PWM signala upotrebom funkcije **softPwmCreate().**

Sledeći deo se odnosi na dobijanje feed-a sa kamere. Upotrebom QcameraInfo klase u qDebug prostor izlistavamo dostupne kamere na našem Rpi računaru. Nakon toga sledi inicijalizacija kamere kreiranjem QCamera objekta za podrazumevanu kameru povezanu na RPI i dodeljuje ga M_Camera pokazivaču. Nakon toga, po sličnom principu kreiramo objekat QCameraViewfinder koji dodeljujemo M_Viewfinder pokazivaču. Nakon toga, naš kamera-feed se povezuje na verticalLayout widget na GUI-u, i povezivanje kamere sa viewfinder-om. Na kraju se vrši podešavanje rezolucije upotrebom klase QcameraViewfinderSettings kao i provera podržane rezolucije.

Na kraju je prikazan tajmer koji će biti upotrebljen od strane PIR senzora.

```
57  Dialog::~Dialog()
58  {
59      delete ui;
60  }
61  void Dialog::moveSG90(int pwmValue)
62  {
63      softPwmWrite(SG90,pwmValue);
64      delay(200);
65      softPwmWrite(SG90,0);
66
67  }
68
69  void Dialog::start_Camera()
70  {
71      qDebug() << "Starting camera...";
72      M_Camera->start();
73      qDebug() << "Camera state: " << M_Camera->state();
74  }
75
76  void Dialog::stop_Camera()
77  {
78      qDebug() << "Stopping camera...";
79      M_Camera->stop();
80      qDebug() << "Camera state: " << M_Camera->state();
81  }
82
83  void Dialog::motion_detected()
84  {
85
86
87
88      if(ui->radioButton_2->isChecked()){
89      if(digitalRead(PIR)==HIGH){
90          if (M_Camera->state() != QCamera::ActiveState) {
91              start_Camera();
92
93              QTimer::singleShot(20000, this, &Dialog::stop_Camera);
94          }
95      }
96      }
97
98  }
99
100
```

Slika 6 - source code drugi deo

Na slici 6 se nalazi funkcija za pomeranje PWM-a. U zavisnosti od pritisnutog tastera, vrednost promenljive pwmValue se smanjuje ili povećava, nakon čega se poziva funkcija moveSG90, te se motor pomera na vrednost datu tom promenljivom.

Nakon toga kreirane su funkcije za zaustavljanje i pokretanje kamere.

Funkcija **motion_detected** je funkcija koja čita vrednost sa PIR sentora. U slučaju da je senzor detektovao pokret (daje visoku vrednost) poziva se funkcija za pokretanje kamere. Nakon toga tajmer kreće da broji 20 sekundi, nakon čega će feed biti prekinut, a senzor ponovo počinje da ”traga” za pokretom. Funkcija motion_detected je aktivna samo ako je *radio_button* na GUI-u selektovan.

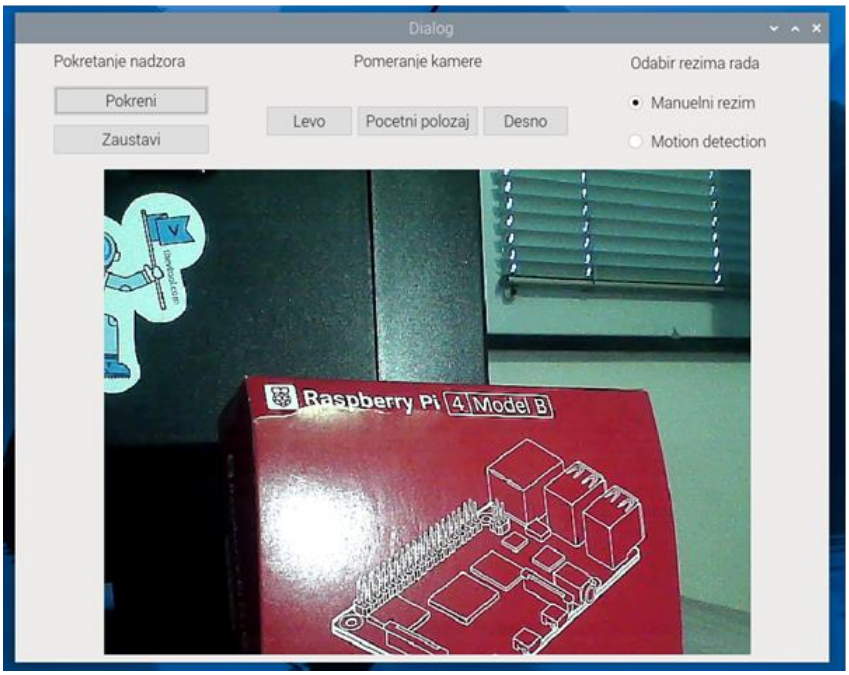
```
101  void Dialog::on_pushButton_Start_clicked()
102  {
103      start_Camera();
104  }
105
106  void Dialog::on_pushButton_Stop_clicked()
107  {
108      stop_Camera();
109  }
110
111  void Dialog::on_radioButton_2_clicked()
112  {
113     ui->pushButton_Start->setDisabled(true);
114     ui->pushButton_Stop->setDisabled(true);
115     stop_Camera();
116  }
117  void Dialog::on_radioButton_clicked()
118  {
119     ui->pushButton_Start->setDisabled(false);
120     ui->pushButton_Stop->setDisabled(false);
121  }
122
123
124  void Dialog::on_motor_levo_clicked()
125  {
126      currentPosition+=1;
127      if(currentPosition>=25)
128          currentPosition=25;
129      qDebug()<<"Trenutna pozivija:  " <<currentPosition;
130      moveSG90(currentPosition);
131  }
132
133
134  void Dialog::on_pushButton_2_clicked()
135  {
136      currentPosition-=1;
137      if(currentPosition<=4)
138          currentPosition=4;
139      qDebug()<<"Trenutna pozivija:  " <<currentPosition;
140      moveSG90(currentPosition);
141  }
142
143
144  void Dialog::on_pushButton_0_clicked()
145  {
146      currentPosition=15;
147      qDebug()<<"Trenutna pozicija: " <<currentPosition;
148      moveSG90(currentPosition);
149  }
```

Slika 7 - source code treći deo

Na slici 7 se nalaze slotovi elementa sa GUI-a. U slučaju da je radioButton_2 pritisnut, nije moguće manuelno pokretati kameru, te su dugmići za te funkcije ”ugašene”. U slučaju pomeranja motora vrši se ograničenje kako vrednost pwm-a ne bi izašla iz operativnog opsega motora.

4.GUI i demonstracija

Na slici 8 prikazan je izgled GUI-a, koji ima 2 segmenta : segment za komande koji zauzima gornji deo interfejsa i segment za prikaz kamere. Na levoj strani nalaze se dugmići za pokretanje i zaustavljanje nadzora. U centralnom delu nalaze se 3 dugmeta, za skretanje u levo, za skretanje u desno i između njih za vraćanje na početni položaj. Sa desne strane nalaze se 2 radio dugmeta pomoću kojih se vrši odabir između mauelnog i automatskog režima rada.



Slika 8 – Grafički interfejs

Na slici 9 prikazana je „maketa" koja je upotrebljena za testiranje i demonstraciju ovog projekta.

Slika 9 – maketa projekta

5. Zaključak

Nakon realizacije projekta, uspešno je ostvaren cilj, te ovaj sistem pruža korisniku intuitivan način upotrebe nadzornog sistema, realizovan na efikasan i jednostavan način.

Ovakva realizacija ostavlja mesta za dalja poboljšanja u budućnosti, kako softverska tako i hardverska.