Софийски университет „Св. Климент Охридски“

Факултет по Математика и Информатика

Курсов проект

Вградени Системи

На тема

Разработване на вградена система, управлявана от смартфон

Изготвил :.....

/Щилиян Александров Узунов/

Проверил:.....

/проф. Васил Георгиев/

Същност на системата

Системата се състои от 2 основни компонента – вградена система с 2 двигателя и 2 гуми, която може да се предвижва по гладки повърхности, и мобилно приложение през което потребителя изпраща команди на вградената система.

Основни компоненти във вградената система са – Ардуино микроконтролер, JY-MCU Bluetooth модул за комуникация, 2 Мотора (и 2 гуми), захранващ модул, L293D интегрална схема за управление на моторите.

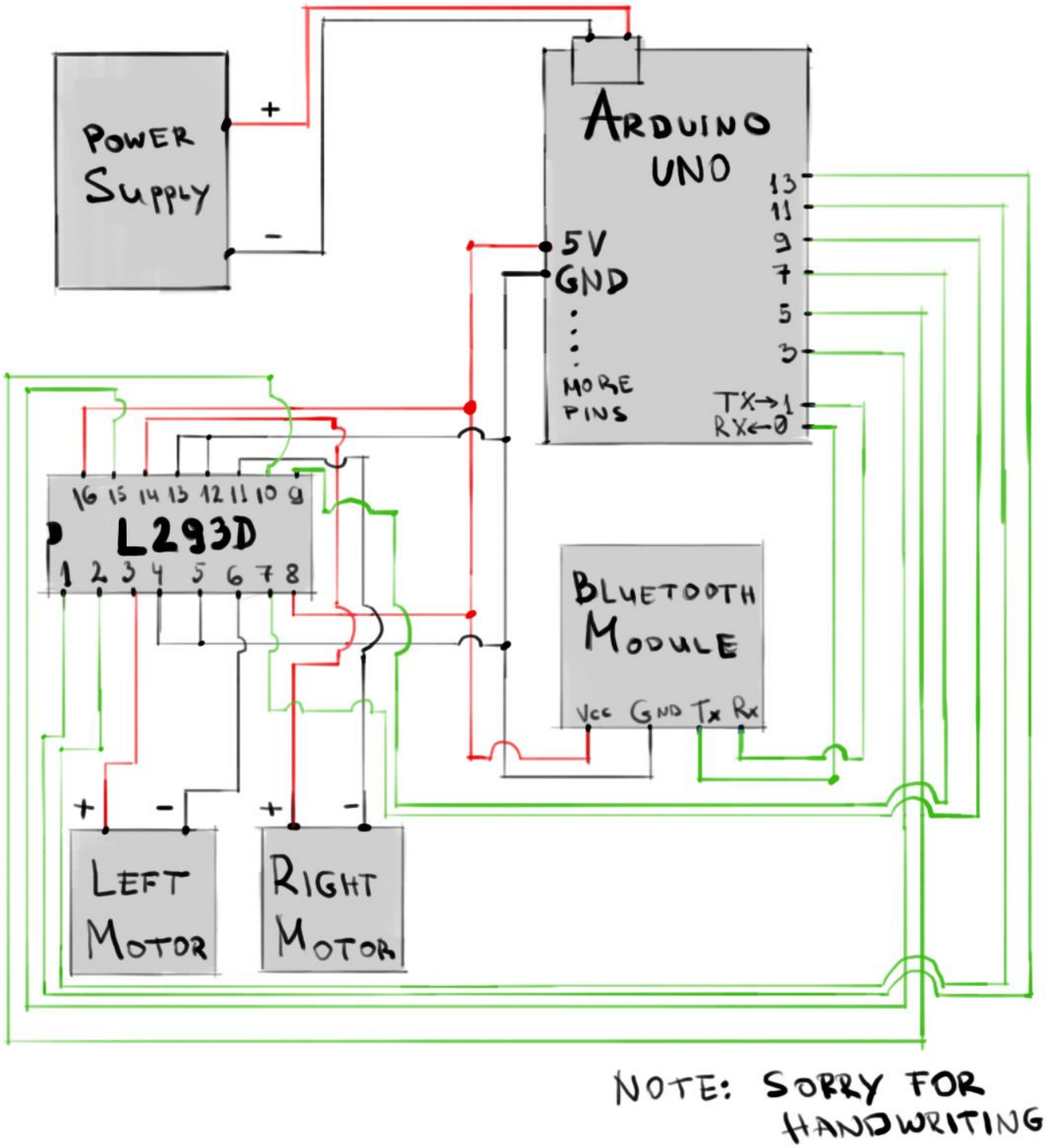
Мобилното приложение е Android базирано, използвният програмен език е JAVA. Използвайки Bluetooth-а на smartphone-а, мобилното приложение успява да осъществи контакт със вградената система.

Поддържат се 4 основни движения – движение напред, назад, въртене на ляво и въртене на дясно.

Мотивация

Избрах да реализирам следната система, защото за нейното изграждане са нужни комбиниране на множество различни компоненти, проектиране на електрическа верига, безжична комуникация, имплементиране на мобилно приложение, и т.н. Всички тези неща са нови за мен, и чрез изпълнението на проекта получих много нови знания.

Електрическа схема на вградената система



Функция на отделните компоненти

- Arduino – Микроконтролера се използва за обработване на информацията получена от смартфона. Захранването идва от захранващият модул. Платката има около 25-30 пина, използваните от тях са показани на схемата.
    - През пинове RX и TX се получава информацията от Bluetooth модула
    - Пинове 3, 5, 7 (Цифрови) служат за управление на десния мотор
    - Пинове 9, 11, 13 (Цифрови) служат за управление на левият мотор
    - 5V пин дава захранване на останалите компоненти
    - GND пин служи за заземяване

  Логиката вградена в микроконтролера можете да видите по-долу в сорс кода.

- Bluetooth модул – служи за комуникация със смартфона. Има 4 пина
    - VCC – за захранване
    - GND – за заземяване
    - RX – за получаване
    - TX – за изпращане

- L293D – Интегрална схема служеща за управление на моторите. Използването на тази схема е нужно, тъй като желаем да можем да придвижваме количката назад и напред. За целта – трябва да можем да управляваме всеки от двигателите напред, и назад. DC Двигателите се въртят в посока от + към - За обръщане на посоката на въртене е необходимо да се смени полярноста на двата кабела на двигателя. Тъй като не желаем да извършваме смяната механично (чрез размяна на пиновете) – използваме L293D интегрална схема. За всеки от двигателите имаме по 3 контролни сигнала, съответно :

    - Enable – за включване/изключване на двигателя
    - Input1 – кодов сигнал1
    - Input2 – кодов сигнал2

  Следната таблица на истинност показва при какви входни сигнали, какво поведение на мотора можем да очакваме :

Enable	Input1	Input2	Function
High	Low	High	Turn clockwise
High	High	Low	Turn anti-clockwise
High	Low	Low	Stop
High	High	High	Stop
Low	Not applicable	Not applicable	Stop

Пинове 4, 5, 12 и 13 – са заземяване. 16 и 8 – захранване. Интегралната схема може да се използва и с използване на външен (по-силен) източник на ток, той се включва на пин 8. В настоящия случай, такъв източник не е използван.

Сорс код на микроконтролера

Основен компонент при този модел на работа на контролера е структурата motorSignals която съдържа в себе си 3 сигнала (Enable, Input1 и Input2) в която се запазва състоянието на сигналите, което трябва да се изпрати към моторите. Състоянието се променя в зависимост от полученото съобщение от серийният порт(Bluetooth модула). Имаме 5 възможни съобщения – F(напред) B(назад) L(на ляво) R(на дясно).

```
 /* Represents the control signals
  * needed to control the motor
  */
 struct motorSignals {
   int enable;
   int input1;
   int input2;
 } left_mSignals, right_mSignals;

char message; // variable to receive data from the serial port

//Pin Mappings for left motor
int leftEnablePin = 13;
int leftInp1Pin = 11;
int leftInp2Pin = 9;

//Pin Mappings for right motor
int rightEnablePin = 7;
int rightInp1Pin = 5;
int rightInp2Pin = 3;
```

```
void turnOffMotors()
{
  left_mSignals.enable = LOW;
  right_mSignals.enable = LOW;
}

void setMoveForward()
{
  left_mSignals.enable = HIGH;
  left_mSignals.input1 = HIGH;
  left_mSignals.input2 = LOW;

  right_mSignals.enable = HIGH;
  right_mSignals.input1 = LOW;
  right_mSignals.input2 = HIGH;
}

void setMoveBackward()
{
  left_mSignals.enable = HIGH;
  left_mSignals.input1 = LOW;
  left_mSignals.input2 = HIGH;

  right_mSignals.enable = HIGH;
  right_mSignals.input1 = HIGH;
  right_mSignals.input2 = LOW;
}

void setTurnLeft()
{
  left_mSignals.enable = HIGH;
  left_mSignals.input1 = HIGH;
  left_mSignals.input2 = LOW;

  right_mSignals.enable = HIGH;
  right_mSignals.input1 = HIGH;
  right_mSignals.input2 = LOW;
}

void setTurnRight()
{
  left_mSignals.enable = HIGH;
  left_mSignals.input1 = LOW;
  left_mSignals.input2 = HIGH;

  right_mSignals.enable = HIGH;
```

```
  right_mSignals.input1 = LOW;
  right_mSignals.input2 = HIGH;
}

void setup() {
  message = 'X';
  pinMode(leftEnablePin, OUTPUT);
  pinMode(leftInp1Pin, OUTPUT);
  pinMode(leftInp2Pin, OUTPUT);
  pinMode(rightEnablePin, OUTPUT);
  pinMode(rightInp1Pin, OUTPUT);
  pinMode(rightInp2Pin, OUTPUT);

  Serial.begin(9600);       // start serial communication at 9600bps
}


/* Main Loop */

void loop() {
  if( Serial.available() > 0 )       // if data is available to read
  {
    message = Serial.read();         // read it and store it in 'val'

    switch (message) {
      case 'F':
         //Serial.println(message);
         setMoveForward();
         break;
      case 'B':
        //Serial.println(message);
        setMoveBackward();
        break;
      case 'L':
        //Serial.println(message);
        setTurnLeft();
        break;
      case 'R':
        //Serial.println(message);
        setTurnRight();
        break;
      case 'X':
        //Serial.println(message);
        turnOffMotors();
        break;
    }
    delay(1);//Delay between reads for stability
  }
```

```
  digitalWrite(leftEnablePin, left_mSignals.enable);
  digitalWrite(leftInp1Pin, left_mSignals.input1);
  digitalWrite(leftInp2Pin, left_mSignals.input2);
  digitalWrite(rightEnablePin, right_mSignals.enable);
  digitalWrite(rightInp1Pin, right_mSignals.input1);
  digitalWrite(rightInp2Pin, right_mSignals.input2);
}
```

Сорс код на мобилното приложение

Class – BluetoothConnector

Класът се използва за осъществяване на връзка със други Bluetooth устройства. За целта са имплементирани методи които да търсят във вече сдвоените (pair-нати) към телефона устройства. Връзката (която представлява отваряне на Bluetooth Socket) се осъществява към Pair-нато устройство, чрез метод който поема име на устройството като входен параметър. Пибличният достъп до входният и изходен поток на класа дава възможност на класът клиент да пише и чете във/от Socket-а.

```
package com.example.shtiliyan.arduinocartcontroller;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;

import java.io.IOException;
import java.util.HashMap;
import java.util.Set;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

/**
 * Created by mitko on 6/26/2014.
 */
public class BluetoothConnector {

    BluetoothAdapter mBluetoothAdapter;
    //Съхранява списъка с пернати устройства
    public static HashMap<String, String> mPairedDevices=new HashMap<String,String>();

    //потоци
    protected BluetoothDevice Device;
    protected BluetoothSocket Socket;
    OutputStream OutputStream = null;
    InputStream InputStream = null;
```

```
private BluetoothDevice CurrentDevice;

//Стартира Bluetooth модулът на устройството ако случайно не е пуснат.
//Връща true или false в зависимост дали е успяло
public boolean enableBluetooth() {

    try {
        if (mBluetoothAdapter == null) {
            mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        }

        if (!mBluetoothAdapter.isEnabled()) {
            mBluetoothAdapter.enable();

        }
        return true;
    } catch (Exception e) {
        return false;
    }
}

public boolean isEnabled(){
    return mBluetoothAdapter.isEnabled();
}

//Спира Bluetooth модулът на устройството ако случайно е
//пуснат.
//Връща true или false в зависимост дали е успяло
public boolean disableBluetooth() {
    try {
        if (mBluetoothAdapter.isEnabled()) {
            mBluetoothAdapter.disable();

        }

        return true;
    } catch (Exception e) {
        return false;
    }
}
//Намира всички пернати устройства Записва ги в локален списък за класа
public void findPairedDivices()
{//Записва всички пернати устройства адаптера
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
```

```
            }

            HashMap<String,String> map;
            mPairedDevices=new HashMap<String,String>();
            if (pairedDevices.size() > 0) {
                for (BluetoothDevice device : pairedDevices) {
                    mPairedDevices.put(device.getName() ,device.getAddress());
                }
            }
        }
        //Намира всички пернати устройства и се свързва към устройството с
        // име deviceName, ако не успее връща съобщение за грешка което после
        // показваме на потребителя като балонче(Intent)
        /*
        public String FindPairedTargetAndConnect(String deviceName)
        {
            String DeviceID;
            if(mPairedDevices.containsKey(deviceName))
            {
                DeviceID= mPairedDevices.get(deviceName);
                if(ConnectToDevice(DeviceID))
                {

                    return "Успешно свързване с устройство: "+deviceName;

                }
                else
                {

                    return "Неуспех при свързване с устройство "+deviceName;

                }

            }
            else
            {

                return "Не е открито устройсто с име: "+deviceName;
            }

        }*/

        //отваря конекция към точно упределено устройство
        public boolean openConnection(String Device)
        {
            String address = mPairedDevices.get(Device);
            CurrentDevice = mBluetoothAdapter.getRemoteDevice(address.toUpperCase());
            try {//Тука се генерира UUID(ид-то е някакво от интернет намерено)
                Socket=CurrentDevice.createRfcommSocketToServiceRecord(UUID.fromString("00001101-
0000-1000-8000-00805F9B34FB"));
            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
            try {
                Socket.connect();
            } catch (IOException e) {
                e.printStackTrace();
                return false;
            }

            try {
                OutputStream = Socket.getOutputStream();
            } catch (IOException e) {
                e.printStackTrace();
                return false;

            }

            try {
                InputStream=Socket.getInputStream();
            } catch (IOException e) {
                e.printStackTrace();
                return false;
            }

            return true;
        }
        //Вътрешен метод за отваряне на конекция. Ползва се от
        // openConnection() и FindPairedTargetAndConnect()
        /*
        private boolean ConnectToDevice(String deviceID) {
            return  openConnection(deviceID);
        }
        */

        public boolean isConnected()
        {
            return Socket.isConnected();
        }
    }
```

Class – ConnectActivity

В този клас е имплементирана логиката на потребителският интерфейс. В UI-а има 5 бутона. Бутон Connect – свързва телефона със Bluetooth модула на вградената система. За целта той използва инстанция от класа BluetoothConnector. След свързване се инициализирах другите 4 бутона, които представляват джойстика за управление на количката. При настъпване на събитие „action_down” всеки бутон изпраща съответната команда за движение. При настъпване на събитие “action_up” всички бутони изпращат съобщение ‚X‘ за стоп на моторите.

```
package com.example.shtiliyan.arduinocartcontroller;
```

```
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.MotionEvent;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.os.Build;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

import java.io.IOException;


public class ConnectActivity extends ActionBarActivity {

    private BluetoothConnector btConnector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        btConnector = new BluetoothConnector();
        btConnector.enableBluetooth();
        btConnector.findPairedDivices();

        setContentView(R.layout.activity_connect);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new PlaceholderFragment())
                    .commit();
        }


    }

    private void initForwardButton(final ImageButton button) {
        button.setOnTouchListener(new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                try {
                    if(event.getAction() == MotionEvent.ACTION_DOWN) {
                        btConnector.OutputStream.write("F".getBytes());
                    }
                    if(event.getAction() == MotionEvent.ACTION_UP) {
```

```
                    btConnector.OutputStream.write("X".getBytes());
                }
            } catch (IOException e) {
                return false;
            }
            return true;
        }
    });
}

private void initBackwardButton(ImageButton button) {
    button.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            try {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    btConnector.OutputStream.write("B".getBytes());
                }
                if(event.getAction() == MotionEvent.ACTION_UP) {
                    btConnector.OutputStream.write("X".getBytes());
                }
            } catch (IOException e) {
                return false;
            }
            return true;
        }
    });
}

private void initTurnLeftButton(ImageButton button) {
    button.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            try {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    btConnector.OutputStream.write("L".getBytes());
                }
                if(event.getAction() == MotionEvent.ACTION_UP) {
                    btConnector.OutputStream.write("X".getBytes());
                }
            } catch (IOException e) {
                return false;
            }
            return true;
        }
    });
}

private void initTurnRightButton(ImageButton button) {
    button.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            try {
```

```
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    btConnector.OutputStream.write("R".getBytes());
                }
                if(event.getAction() == MotionEvent.ACTION_UP) {
                    btConnector.OutputStream.write("X".getBytes());
                }
                } catch (IOException e) {
                    return false;
                }
                return true;
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_connect, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {

        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_connect, container, false);

            return rootView;
        }
    }
```

```
    public void connectButtonClick(View v)
    {
        btConnector.openConnection("HC-06");
        Button button = (Button)v.findViewById(R.id.connectButton);
        button.setClickable(false);
        button.setEnabled(false);

        initForwardButton((ImageButton) findViewById(R.id.forwardButton));
        initBackwardButton((ImageButton) findViewById(R.id.backwardButton));
        initTurnLeftButton((ImageButton) findViewById(R.id.turnLeftButton));
        initTurnRightButton((ImageButton) findViewById(R.id.turnRightButton));
    }
}
```

Предложения за бъдещо развитие

- Да се добави сензор за разстояние, спрямо който да се ограничава движението напред на количката. При засечена голяма близост на даден предмет – движението напред да се прекрати. Това елиминира възможноста от сблъсъци.
- Да се добави възможност количката да се управлява през жироскопа на телефона. Това би направило потребителската интеракция със количката по-интуитивна.
- Да се подобрят връзките между някой от кабелите, защото са залепени с ХАРТИЕНО ТИКСО! (Виж модела)