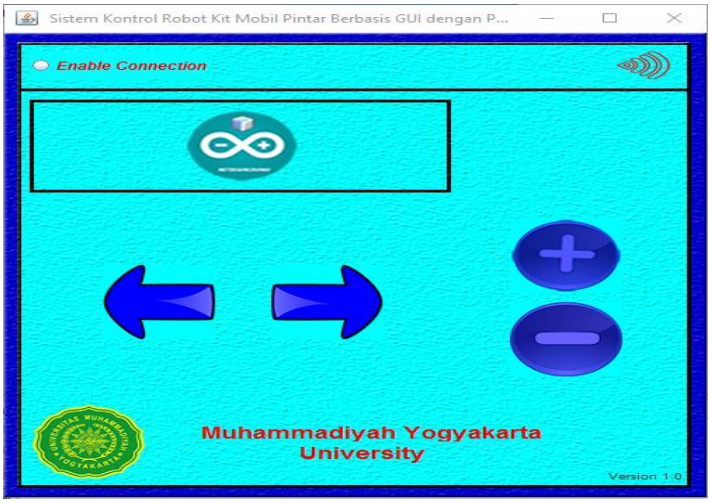
BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

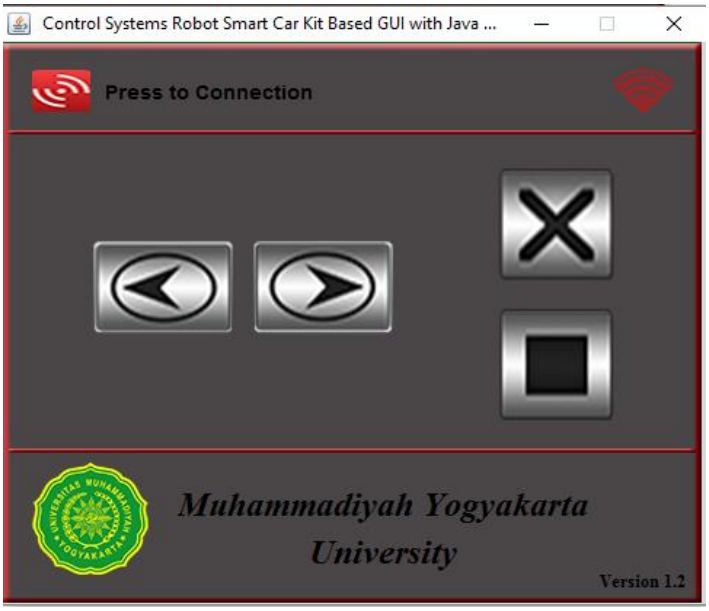
4.1.1 Tampilan Program

Sesuai dengan metode penelitian pada bab III B, penelitian ini menghasilkan 3 buah desain *Graphical User Interface* (GUI) seperti yang ditunjukkan oleh gambar 4.1, gambar 4.2, dan gambar 4.3.



Gambar 4.1. Desain GUI version 1.0

Gambar 4.2. Desain GUI version 1.1



Gambar 4.3. Desain GUI version 1.2

Desain *Graphical User Interface* pada gambar 4.1, gambar 4.2, gambar 4.3 diatas dirancang berdasarkan metode kombinasi warna terbaik dan kombinasi warna terjelek. Kombinasi warna pada sebuah desain GUI berfungsi untuk menghindari mata dari kelelahan, terlihat lebih menarik, dan memudahkan *user* dalam memahami GUI. Berikut tabel kombinasi warna terbaik dan kombinasi warna terjelek dalam mendesain GUI yang ditunjukkan pada gambar 4.4 dan gambar 4.5.

.

Latar Belakang	Garis Tipis dan Teks	Garis Tebal dan Teks
Putih	Biru(94%), Hitam(63%), Merah(25%)	Hitam(69%), Biru(63%), Merah(31%)
Hitam	Putih(75%), Kuning(63%)	Kuning(69%), Putih(59%), Hijau(25%)
Merah	Kuning(75%), Putih(56%), Hitam(44%)	Hitam(50%), Kuning(44%), Putih(44%), Cyan(31%)
Hijau	Hitam(100%), Biru(56%), Merah(25%)	Hitam(69%), Merah(63%), Biru(31%)
Biru	Putih(81%), Kuning(50%), Cyan(25%)	Kuning(38%), Magenta(31%), Hitam(31%), Cyan(31%), Putih(25%)
Cyan	Biru(69%), Hitam(56%), Merah(37%)	Merah(56%), Biru(%), Hitam(44%), Magenta(25%)
Magenta	Hitam(63%), Putih(56%), Biru(44%)	Biru(50%), Hitam(44%), Kuning(25%)
Kuning	Merah(63%), Biru(63%), Hitam(56%)	Merah(75%), Biru(63%), Hitam(50%)

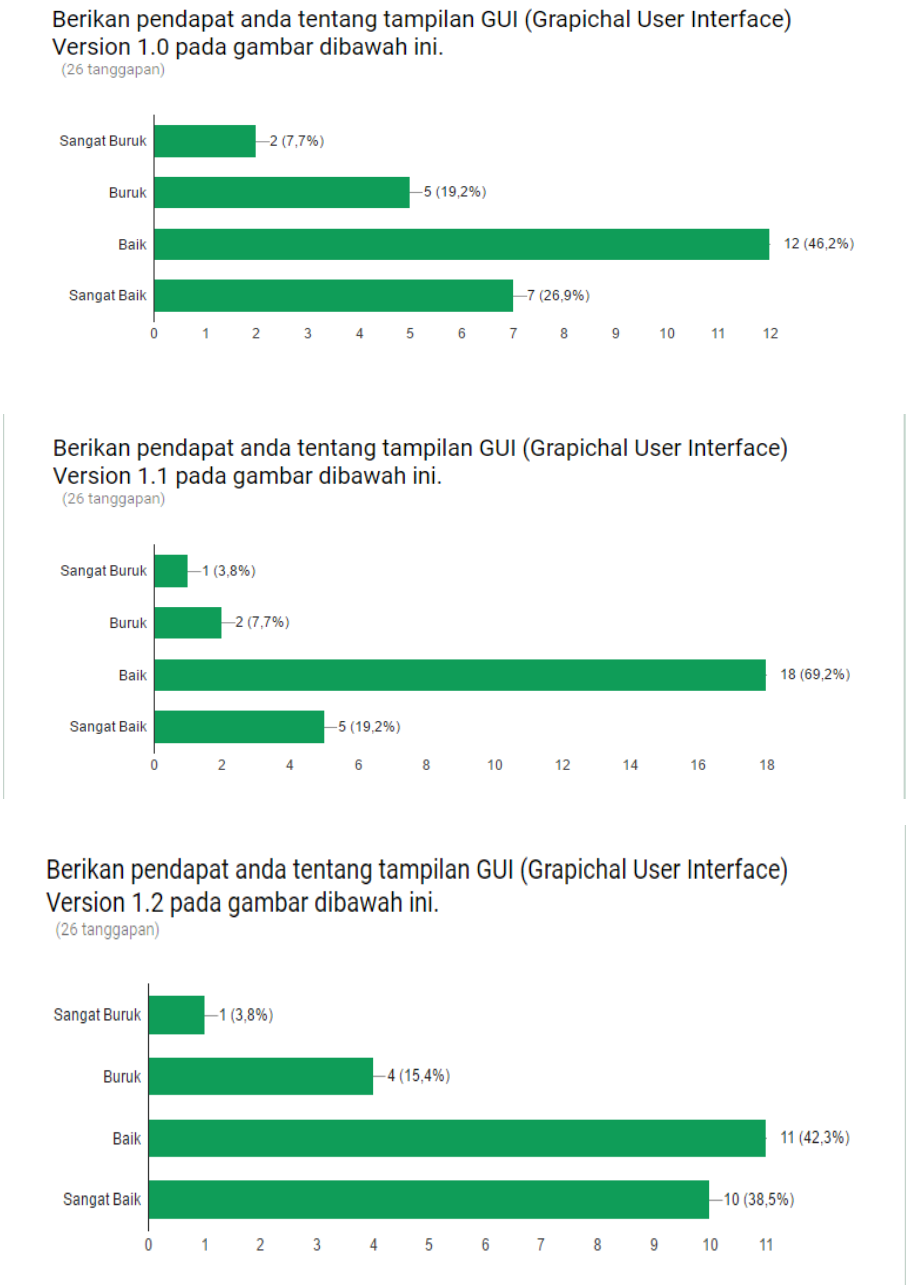
Gambar 4.4. Kombinasi warna terbaik

Latar Belakang	Garis Tipis dan Teks	Garis Tebal dan Teks
Putih	Kuning(100%), Cyan(94%)	Kuning(94%), Cyan(75%)
Hitam	Biru(89%), Merah(44%), Magenta(25%)	Biru(81%), Magenta(31%)
Merah	Magenta(81%), Biru(44%), Hijau dan Cyan(21%)	Biru(81%),Magenta(31%)
Hijau	Cyan(81%),Magenta(50%), Kuning(37%)	Cyan(81%),Magenta & Kuning(44%)
Biru	Hijau(62%), Merah& Hitam(37%)	Hijau(44%), Merah& Hitam(31%)
Cyan	Hitam(81%), Kuning(75%), Putih(31%)	Kuning(69%), Hijau(62%), Putih(56%)
Magenta	Hijau(75%), Merah(56%), Cyan(44%)	Cyan(81%), Hijau(69%), Merah(44%)
Kuning	Putih dan Cyan (81%)	Putih(81%), Cyan(56%), Hijau (25%)

Gambar 4.5. Kombinasi warna terjelek

Desain GUI yang digunakan pada penelitian adalah desain GUI versi 1.2. Desain ini dipilih berdasarkan hasil dari kuisioner yang dipilih oleh beberapa peserta kuisoner. Pada desain GUI versi 1.0 dua orang memilih sangat buruk, lima orang memilih buruk, dua belas orang memilih baik dan tujuh orang memilih sangat baik. Pada desain GUI versi 1.1 satu orang memilih sangat buruk, dua orang

memilih buruk, delapan belas orang memilih baik dan lima orang memilih sangat baik. Pada desain GUI versi 1.2 satu orang memilih sangat buruk, empat orang memilih buruk, sebelas orang memilih baik dan sepuluh orang memilih sangat baik. Gambar 4.6 menunjukkan hasil kuisioner yang sudah dibuat.



Gambar 4.6. Hasil Kuisoner desain GUI

4.1.2 Tampilan Kit Robot Mobil Pintar 4WD

Kit robot mobil pintar 4WD ini dirancang dengan beberapa komponen, seperti yang ditunjukkan oleh gambar 4.7 yaitu :

a. Arduino Mega 2560.
b. *Driver L298N dual H-Bridge*.
c. 4 buah motor DC.
d. Modul *Radio 3DR Telemetry 433MHz Receivier*.
e. *Battery* 3S, 25C, 1800mah.
f. 1 buah *switch*.
g. Beberapa kabel *jumper male to male*.

Gambar 4.7. Tampilan Kit Mobil Pintar 4WD

4.2 Pembahasan

4.2.1 *Source Code* Pada GUI

Rancangan GUI pada penelitian ini dibuat berdasarkan *JFrame Form* yang terdapat pada *software* NetBeans dan *source code* di bawah ini :

```
package GUI_White;

import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.awt.Color;
import static java.awt.image.ImageObserver.ERROR;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Enumeration;
import java.util.TooManyListenersException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

/**
 *
 * @author Sony-Vaio
 */
public class Frame_GUI extends javax.swing.JFrame
implements SerialPortEventListener {

    private static final String FORWARD = "F";
    private static final String REVERSE = "B";
    private static final String RIGHT   = "R";
    private static final String LEFT    = "L";
    private static final String STOP    = "S";
    /**
     * Creates new form Frame_GUI
     */

    private OutputStream output = null;
    private InputStream input = null;
    SerialPort serialPort;
    private final String PORT = "COM3";
    private static final int TIMEOUT = 2000;
    private static final int DATA_RATE = 57600;
```

```
public Frame_GUI() {
        initComponents();
        butwifi.setVisible(true);
        butoff.setVisible(true);
        setTitle(" Control Systems Robot Smart Car
Kit Based GUI with Java Programming. ");
    }

    public void inisialKoneksi() {
        CommPortIdentifier portID = null;
        Enumeration portEnum =
CommPortIdentifier.getPortIdentifiers();

        while (portEnum.hasMoreElements()) {
            CommPortIdentifier actualPortID =
(CommPortIdentifier) portEnum.nextElement();
            if
(PORT.equals(actualPortID.getName())) {
                portID = actualPortID;
                break;
                }
        }

        if (portID == null) {
            mostrarError("error");
            System.exit(ERROR);
        }

        try {
            serialPort = (SerialPort)
portID.open(this.getClass().getName(), TIMEOUT);

serialPort.setSerialPortParams(DATA_RATE,
SerialPort.DATABITS_8, SerialPort.STOPBITS_2,
SerialPort.PARITY_NONE);

            output = serialPort.getOutputStream();
        } catch (Exception e) {
            mostrarError(e.getMessage());
            System.exit(ERROR);
        }


    try {
            input = serialPort.getInputStream();
        } catch (IOException ex) {

Logger.getLogger(Frame_GUI.class.getName()).log(Lev
el.SEVERE, null, ex);
        }


        try {
            serialPort.addEventListener(this);
            serialPort.notifyOnDataAvailable(true);
        } catch (TooManyListenersException ex) {
```

```
Logger.getLogger(Frame_GUI.class.getName()).log(Lev
el.SEVERE, null, ex);
        }
    }


    @Override
    public void serialEvent(SerialPortEvent spe) {
        if (spe.getEventType() ==
SerialPortEvent.DATA_AVAILABLE) {
            byte[] readBuffer = new byte[20];
            try {
                int numBytes = 1;

                while (input.available() >0) {
                    numBytes =
input.read(readBuffer);
    }
                  //areasensor.append(new
String(readBuffer, 0, numBytes, "us-ascii"));
            } catch (IOException e) {
                System.out.println(e);
            }
        }
    }

     private void kirimData(String data) {
        try {
            output.write(data.getBytes());
        } catch (Exception e) {
            mostrarError("Tidak Konek");
            System.exit(ERROR);
        }
    }


     public void closeSerial(){
        if(serialPort!=null){
            serialPort.removeEventListener();
            serialPort.close();
            new Frame_GUI().setVisible(false);
            JOptionPane.showMessageDialog(this, "no
koneksi!\n");
        }
    }


     public void mostrarError(String mensaje) {
     JOptionPane.showMessageDialog(this, mensaje,
"ERROR", JOptionPane.ERROR_MESSAGE);
     }
```

```
@SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        mundur = new javax.swing.JButton();
        maju = new javax.swing.JButton();
        kiri = new javax.swing.JButton();
        kanan = new javax.swing.JButton();
        butwifi = new javax.swing.JToggleButton();
        txtwifi = new javax.swing.JLabel();
        lblwifi = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        butoff = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstan
ts.EXIT_ON_CLOSE);
        setMaximumSize(new java.awt.Dimension(490,
430));
        setMinimumSize(new java.awt.Dimension(490,
430));
        setPreferredSize(new
java.awt.Dimension(490, 430));
        setResizable(false);
        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

        mundur.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/mun
durx.png"))); // NOI18N
        mundur.setMaximumSize(new
java.awt.Dimension(50, 50));
        mundur.setMinimumSize(new
java.awt.Dimension(50, 50));
        mundur.setPreferredSize(new
java.awt.Dimension(50, 50));
        mundur.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                mundurMousePressed(evt);
            }
        });
        mundur.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                mundurActionPerformed(evt);
            }
        });
```

```
getContentPane().add(mundur, new
org.netbeans.lib.awtextra.AbsoluteConstraints(340,
190, 80, 80));

        maju.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/maj
ux.png"))); // NOI18N
        maju.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                majuMousePressed(evt);
            }
        });
        maju.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                majuActionPerformed(evt);
            }
        });
        getContentPane().add(maju, new
org.netbeans.lib.awtextra.AbsoluteConstraints(340,
90, 80, 80));

        kiri.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/kir
ix.png"))); // NOI18N
        kiri.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                kiriMousePressed(evt);
            }
        });
        kiri.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                kiriActionPerformed(evt);
            }
        });
        getContentPane().add(kiri, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60,
140, 100, 70));

        kanan.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/kan
anx.png"))); // NOI18N
        kanan.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                kananMousePressed(evt);
            }
        });
```

```
kanan.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                kananActionPerformed(evt);
            }
        });
        getContentPane().add(kanan, new
org.netbeans.lib.awtextra.AbsoluteConstraints(170,
140, 100, 70));

        butwifi.setBackground(new
java.awt.Color(71, 71, 71));
        butwifi.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ON.
png"))); // NOI18N

butwifi.setBorder(javax.swing.BorderFactory.create
EmptyBorder(1, 1, 1, 1));
        butwifi.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                butwifiActionPerformed(evt);
            }
        });
        getContentPane().add(butwifi, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20,
20, 30, 30));

        txtwifi.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/wif
i-empty-xxl.png"))); // NOI18N
        getContentPane().add(txtwifi, new
org.netbeans.lib.awtextra.AbsoluteConstraints(420,
20, 50, 30));

        lblwifi.setFont(new java.awt.Font("Vani",
1, 14)); // NOI18N
        lblwifi.setText("Press to Connection");
        getContentPane().add(lblwifi, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60,
20, 150, 30));

        jLabel1.setFont(new java.awt.Font("Times
New Roman", 3, 24)); // NOI18N
        jLabel1.setText("Muhammadiyah
Yogyakarta");
        getContentPane().add(jLabel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(120,
310, 410, 40));
```

```
jLabel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Logo
_of_Muhammadiyah_University_of_Yogyakarta.png")));
// NOI18N
        getContentPane().add(jLabel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20,
300, 80, 80));

        jLabel5.setFont(new java.awt.Font("Times
New Roman", 3, 24)); // NOI18N
        jLabel5.setText("University");
        getContentPane().add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(210,
350, 140, -1));

        jLabel6.setFont(new java.awt.Font("Times
New Roman", 1, 12)); // NOI18N
        jLabel6.setText("Version 1.2");
        getContentPane().add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(410,
370, 80, 30));

        butoff.setBackground(new java.awt.Color(71,
71, 71));
        butoff.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/OFF.
png"))); // NOI18N
        butoff.setBorder(null);
        butoff.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                butoffActionPerformed(evt);
            }
        });
        getContentPane().add(butoff, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20,
20, 30, 30));

        jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/FX4.
jpg"))); // NOI18N
        getContentPane().add(jLabel2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0,
480, 400));

        setBounds(0, 0, 496, 437);
    }// </editor-fold>//GEN-END:initComponents
```

```
private void
majuActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_majuActionPerformed
        kirimData(STOP);
        maju.setIcon(new
ImageIcon("src/majux.png"));

    }//GEN-LAST:event_majuActionPerformed

    private void
majuMousePressed(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_majuMousePressed
        kirimData(FORWARD);
        maju.setIcon(new
ImageIcon("src/majuon.png"));
    }//GEN-LAST:event_majuMousePressed

    private void
mundurActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_mundurActionPerformed
        kirimData(STOP);
        mundur.setIcon(new
ImageIcon("src/mundurx.png"));
    }//GEN-LAST:event_mundurActionPerformed

    private void
mundurMousePressed(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_mundurMousePressed
        kirimData(REVERSE);
        mundur.setIcon(new
ImageIcon("src/munduron.png"));
    }//GEN-LAST:event_mundurMousePressed

    private void
kananActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_kananActionPerformed
        kirimData(STOP);
        kanan.setIcon(new
ImageIcon("src/kananx.png"));
    }//GEN-LAST:event_kananActionPerformed

    private void
kananMousePressed(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_kananMousePressed
        kirimData(RIGHT);
        kanan.setIcon(new
ImageIcon("src/kananon.png"));
    }//GEN-LAST:event_kananMousePressed

    private void
kiriActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_kiriActionPerformed
        kirimData(STOP);
        kiri.setIcon(new
ImageIcon("src/kirix.png"));
```

```
private void
kiriMousePressed(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_kiriMousePressed
        kirimData(LEFT);
        kiri.setIcon(new
ImageIcon("src/kirion.png"));
    }//GEN-LAST:event_kiriMousePressed

    private void
butwifiActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_butwifiActionPerformed
        butwifi.setVisible(false);
        butoff.setVisible(true);
        txtwifi.setIcon(new
ImageIcon("src/baron.png"));
        lblwifi.setText("Connected");
        lblwifi.setForeground(Color.red);
         inisialKoneksi();
    }//GEN-LAST:event_butwifiActionPerformed

    private void
butoffActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_butoffActionPerformed
        closeSerial();
        butwifi.setVisible(true);
        butoff.setVisible(false);
        lblwifi.setText("Disconnected");
        lblwifi.setForeground(Color.black);
        txtwifi.setIcon(new ImageIcon("src/wifi-
empty-xxl.png"));
    }//GEN-LAST:event_butoffActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed"
desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is
not available, stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/
lookandfeel/plaf.html
         */

        //</editor-fold>

        /* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Frame_GUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-
BEGIN:variables
    private javax.swing.JButton butoff;
    private javax.swing.JToggleButton butwifi;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton kanan;
    private javax.swing.JButton kiri;
    private javax.swing.JLabel lblwifi;
    private javax.swing.JButton maju;
    private javax.swing.JButton mundur;
    private javax.swing.JLabel txtwifi;
    // End of variables declaration//GEN-
END:variables
}
```

4.2.1.1 Penjelasan *Source Code* GUI

Pada penelitian ini dibutuhkan beberapa *library* pada saat merancang sebuah GUI, beberapa *library* yang digunakan pada penelitian ini yaitu :

1. RXTXcomm yang berfungsi untuk memanggil kelas – kelas seperti *import gnu.io.CommPortIdentifier; import gnu.io.SerialPort; import gnu.io.SerialPortEvent; import gnu.io.SerialPortEventListener; import java.io.IOException; import java.io.InputStream; import java.io.OutputStream; import java.util.Enumeration;*
2. AbsoluteLayout dan beberapa *library* yang sudah *include* pada JDK 1.8 berfungsi memanggil kelas – kelas seperti *import*

```
java.util.TooManyListenersException; import
java.util.logging.Level; import java.util.logging.Logger;
import javax.swing.ImageIcon; import
javax.swing.JOptionPane; import java.awt.Color;
import static java.awt.image.ImageObserver.ERROR;
```

Setelah memasukkan *library* dan memanggil kelas – kelas yang dibutuhkan selanjutnya masuk ketahap berikutnya yaitu menuliskan fungsi fungsi.

Langkah pertama adalah membuat inisialisasi data yang akan dikirimkan ke Arduino Mega2560 dalam bentuk karakter.

```
private static final String FORWARD = "F";
private static final String REVERSE = "B";
private static final String RIGHT   = "R";
private static final String LEFT    = "L";
private static final String STOP    = "S";
```

Gambar 4.8. Inisialisasi data

Pada gambar 4.8 menunjukkan inisialisasi data dalam bentuk fungsi *String* yang diubah kedalam bentuk kode ASCII yaitu berupa karakter. Karakter inilah yang nantinya dikirimkan ke Arduino Mega 2560.

Langkah kedua adalah menuliskan fungsi – fungsi yang akan digunakan untuk berkomunikasi antara Laptop/PC dengan Arduino Mega 2560.

```
private OutputStream output = null;
private InputStream input = null;
SerialPort serialPort;
private final String PORT = "COM3";
private static final int TIMEOUT = 2000;
private static final int DATA_RATE = 57600;
```

Gambar 4.9. Penulisan fungsi – fungsi komunikasi antara Laptop/PC dengan Arduino Mega 2560

Pada gambar 4.9 menunjukkan untuk dapat berkomunikasi membutuhkan fungsi fungsi yang ada pada gambar di atas. Fungsi – fungsi tersebut disediakan oleh *library* RXTXcomm, sedangkan nilai *baud rate* dan *port* dapat disesuaikan dengan modul radio 3DR yang digunakan.

Langkah ketiga adalah menuliskan kelas inisialisasi koneksi, tujuannya adalah untuk memanggil kembali fungsi – fungsi komunikasi yang ada pada langkah kedua. Pada penulisan inisialisasi koneksi ini memakai kelas *public*, kelas *public* digunakan agar seluruh *kelas/method/attribute* pada kelas ini dapat diakses oleh kelas manapun. Pada gambar 4.10 menunjukkan penulisan program untuk inisialisasi koneksi.

```
public void inisialKoneksi() {
    CommPortIdentifier portID = null;
    Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

    while (portEnum.hasMoreElements()) {
        CommPortIdentifier actualPortID = (CommPortIdentifier) portEnum.nextElement();
        if (PORT.equals(actualPortID.getName())) {
            portID = actualPortID;
            break;
            }
    }
    if (portID == null) {
        mostrarError("error");
        System.exit(ERROR);
    }
    try {
        serialPort = (SerialPort) portID.open(this.getClass().getName(), TIMEOUT);
        serialPort.setSerialPortParams(DATA_RATE, SerialPort.DATABITS_8, SerialPort.STOPBITS_2, SerialPort.PARITY_NONE);
        output = serialPort.getOutputStream();
    } catch (Exception e) {
        mostrarError(e.getMessage());
        System.exit(ERROR);
    }
    try {
        input = serialPort.getInputStream();
    } catch (IOException ex) {
        Logger.getLogger(Frame_GUI.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
    } catch (TooManyListenersException ex) {
        Logger.getLogger(Frame_GUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Gambar 4.10. Kelas inisialisasi koneksi

Langkah keempat adalah menuliskan kelas untuk mengirimkan data, tujuannya untuk mengirimkan data yang sudah di inisialisasikan seperti pada langkah pertama. Pada kelas untuk mengirimkan data memakai kelas *private* yang artinya kelas ini tidak dapat diakses sama sekali oleh kelas lain. Gambar 4.11 menunjuukan penulisan kelas untuk mengirimkan data.

```
private void kirimData(String data) {
    try {
        output.write(data.getBytes());
    } catch (Exception e) {
        mostrarError("Tidak Konek");
        System.exit(ERROR);
    }
}
```

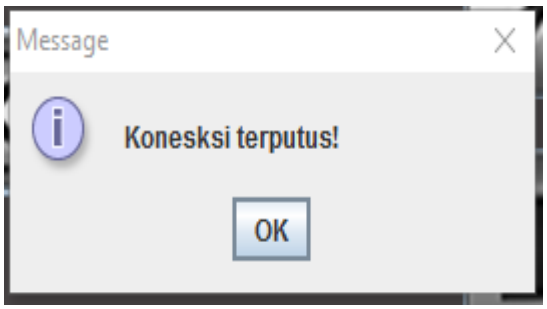
Gambar 4.11. Kelas kirim data

Langkah kelima adalah menuliskan kelas untuk memutuskan koneksi, fungsinya memutus koneksi serial antara laptop dengan Arduino Mega 2560. Pada penulisan kelas memutus koneksi ini memakai kelas *public*, kelas *public* digunakan agar seluruh *kelas/method/attribute* pada kelas ini dapat diakses oleh kelas manapun. Gambar 4.12 menunjukkan penulisan kelas memutus *port* serial.

```
public void closeSerial(){
    if(serialPort!=null){
        serialPort.removeEventListener();
        serialPort.close();
        new Frame_GUI().setVisible(false);
        JOptionPane.showMessageDialog(this, "Koneksi terputus!\n");
    }
}
```

Gambar 4.12. Kelas menutup koneksi serial

Untuk mengetahui koneksi serial sudah terputus atau tidak perlu dibuat notifikasi pemberitahuan dengan memunculkan jendela pesan. Perintah untuk memunculkan notifikasi tersebut adalah kode program pada baris terakhir yang terdapat pada gambar 4.12. Gambar 4.13 menunjukkan jendela notifikasi pemberitahuan koneksi terputus.



Gambar 4.13. Jendela notifikasi koneksi terputus

Langkah keenam adalah menuliskan kelas *error* untuk menutup GUI apabila modul radio 3DR belum dipasang atau tidak ada koneksi sama sekali. Pada penulisan kelas memutus koneksi ini memakai kelas *public*, kelas *public* digunakan agar seluruh *kelas/method/attribute* pada kelas ini dapat diakses oleh kelas manapun. Gambar 4.14 menunjukkan penulisan kelas *error*.

```
public void mostrarError(String mensaje) {
JOptionPane.showMessageDialog(this, mensaje, "ERROR", JOptionPane.ERROR_MESSAGE);
}
```

Gambar 4.14. Penulisan kelas *error*

Langkah terakhir adalah membuat kelas – kelas dan perintah untuk *button* navigasi dan koneksi. Fungsinya untuk menampilkan

button navigasi dan koneksi pada GUI. Beberapa jenis kelas untuk button navigasi dan koneksi yaitu :

1. Kelas *Butto*n Maju

   Button maju ini berfungsi sebagai kelas yang menampilkan *button* maju dan mengirim data karakter "F" dan "S" serta merubah *icon button*. Penulisan kode program untuk kelas *button maju* ditujunkkan pada gambar 4.15.

```
private void majuActionPerformed(java.awt.event.ActionEvent evt) {
    kirimData(STOP);
    maju.setIcon(new ImageIcon("src/majux.png"));

}

private void majuMousePressed(java.awt.event.MouseEvent evt) {
    kirimData(FORWARD);
    maju.setIcon(new ImageIcon("src/majuon.png"));
}
```

Gambar 4.15. Kelas *button* maju

Sedangkan untuk tampilan *icon button* maju ditunjukkan pada gambar 4.16.

Gambar 4.15. *Icon button* maju

2. Kelas *Button* Mundur

Button mundur ini berfungsi sebagai kelas yang menampilkan *button* mundur dan mengirim data karakter "B" dan "S" serta merubah *icon button*. Penulisan kode program untuk kelas *button* mundur ditujunkkan pada gambar 4.17.

```
private void mundurActionPerformed(java.awt.event.ActionEvent evt) {
    kirimData(STOP);
    mundur.setIcon(new ImageIcon("src/mundurx.png"));
}

private void mundurMousePressed(java.awt.event.MouseEvent evt) {
    kirimData(REVERSE);
    mundur.setIcon(new ImageIcon("src/munduron.png"));
}
```

Gambar 4.17. Kelas *button* mundur

Sedangkan untuk tampilan *icon button* mundur ditunjukkan pada gambar 4.18.

Gambar 4.19. *Icon button* mundur

3. Kelas *Button* Kanan

Button kanan ini berfungsi sebagai kelas yang menampilkan *button* kanan dan mengirim data karakter "R" dan "S" serta merubah *icon button*. Penulisan kode program untuk kelas *button* kanan ditujunkkan pada gambar 4.20.

```
private void kananActionPerformed(java.awt.event.ActionEvent evt) {
    kirimData(STOP);
    kanan.setIcon(new ImageIcon("src/kananx.png"));
}

private void kananMousePressed(java.awt.event.MouseEvent evt) {
    kirimData(RIGHT);
    kanan.setIcon(new ImageIcon("src/kananon.png"));
}
```

Gambar 4.20. Kelas *button* kanan

Sedangkan untuk tampilan *icon button* kanan ditunjukkan pada gambar 4.21.

Gambar 4.21. *Icon button* kanan

4. Kelas *Button* Kiri

   Button kiri ini berfungsi sebagai kelas yang menampilkan *button* kiri dan mengirim data karakter “L” dan “S” serta merubah *icon button*. Penulisan kode program untuk kelas *button* kiri ditujunkkan pada gambar 4.22.

```
private void kiriActionPerformed(java.awt.event.ActionEvent evt) {
    kirimData(STOP);
    kiri.setIcon(new ImageIcon("src/kirix.png"));
}

private void kiriMousePressed(java.awt.event.MouseEvent evt) {
    kirimData(LEFT);
    kiri.setIcon(new ImageIcon("src/kirion.png"));
}
```

Gambar 4.22. Kelas *button* kiri

Sedangkan untuk tampilan *icon button* kiri ditunjukkan pada gambar 4.23.

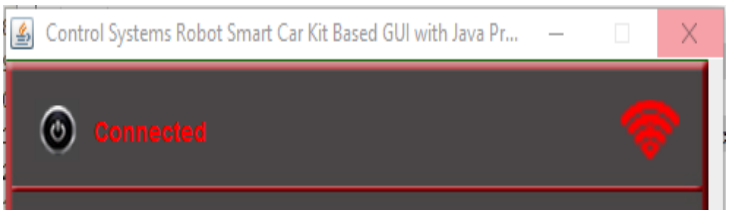
Gambar 4.23. *Icon button* kiri

5. Kelas *Button* Koneksi dan Pemutus Koneksi

   Button koneksi ini berfungsi sebagai pengakses kelas inisialisasi koneksi yang tujuannya mengkoneksikan laptop/PC dengan robot melalui *port* serial yang terdapat pada laptop/PC yang memakai modul *RCTimer Radio Telemetry Kit 433MHz* sebagai medianya serta menampilkan *icon button* koneksi dan *icon* notifikasi berbentuk *bar wifi* penuh. Gambar 4.24 menunjukkan penulisan kelas untuk *button* koneksi dan gambar 4.25 menunjukkan *icon button* koneksi serta notifikasi *bar wifi* penuh.

```
private void butwifiActionPerformed(java.awt.event.ActionEvent evt) {
    butwifi.setVisible(false);
    butoff.setVisible(true);
    txtwifi.setIcon(new ImageIcon("src/baron.png"));
    lblwifi.setText("Connected");
    lblwifi.setForeground(Color.red);
     inisialKoneksi();
}
```

Gambar 4.24. Kelas *button* koneksi

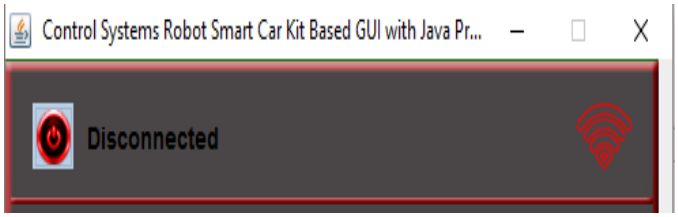


Gambar 4.25. *Icon button* koneksi dan bar wifi

Sedangkan *button* pemutus koneksi berfungsi mengakses kelas *closeSerial* yang ada pada langkah kelima serta menampilkan notifikasi *disconnected* dan *bar wifi* kosong. Gambar 4.26 menunjukkan penulisan kelas button pemutus koneksi dan gambar 4.27 menunjukkan *icon* notifikasi serta *bar wifi* kosong.

```
private void butoffActionPerformed(java.awt.event.ActionEvent evt) {
    closeSerial();
    butwifi.setVisible(true);
    butoff.setVisible(false);
    lblwifi.setText("Disconnected");
    lblwifi.setForeground(Color.black);
    txtwifi.setIcon(new ImageIcon("src/wifi-empty-xxl.png"));
}
```

Gambar 4.26. Kelas button pemutus koneksi



Gambar 4.27. *Icon button* pemutus koneksi dan notifikasi serta bar *wifi*

Pada seluruh kelas *button* navigasi terdapat kelas kirim data dan inisialisasi data yang tujuannya agar karakter huruf yang ada pada inisialisasi data dapat terkirim sesuai dengan fungsi tombol navigasi masing – masing. Seluruh tombol navigasi diberi *event* `mousePressed` dan `actionPerformed`. *Event* `actionPerformed` berfungsi untuk mengirim inisialisasi data karakter "F", "B", "R", dan "L" ketika kursor diletakkan *pada icon button* navigasi tertentu kemudian memberikan aksi dengan cara menekan *button* tersebut melalui klik kiri pada *touchpad* laptop ataupun dengan mouse. *Event* `mousePressed` berfungsi untuk mengirim inisialisasi data karakter "S" ketika *button* dilepas dengan cara melepas klikan pada *touchpad* maupun mouse. Sedangkan fungsi *setIcon* pada tiap kelas adalah untuk merubah icon masing – masing button dan label pada GUI.

4.2.2 Source Code Pada Arduino Mega 2560

Arduino Mega 2560 bertugas sebagai penerima (*receivier*) data dari laptop atau pngirim (*transmitter*). Untuk mengontrol keseluruhan komponen baik *input* maupun *output* sesuai dengan yang dibutuhkan pada penelitian ini, Arduino Mega 2560 di program menggunakan *software* Arduino IDE berdasarkan *source code* di bawah ini :

```
//Initial Receivier Data
char karakter;
String komen;
void setup()
  {
    Serial.begin(57600);
    pinMode(22, OUTPUT);
    pinMode(23, OUTPUT);
    pinMode(24, OUTPUT);
    pinMode(25, OUTPUT);
  }
void loop()
  {
    while (Serial.available()>0)
    {
    karakter= Serial.read();
    komen.concat(karakter);
    delay(10);
    }
//Forward
      if (komen.equals("F") == true)
      {
      digitalWrite(22, HIGH);
      digitalWrite(23, LOW);
      digitalWrite(24, LOW);
      digitalWrite(25, HIGH);
      }
      //Reverse
      if (komen.equals("B")== true)
      {
      digitalWrite(22, LOW);
      digitalWrite(23, HIGH);
      digitalWrite(24, HIGH);
      digitalWrite(25, LOW);
      }
      //Right
      if (komen.equals("R")== true)
      {
      digitalWrite(22, HIGH);
      digitalWrite(23, LOW);
      digitalWrite(24, HIGH);
      digitalWrite(25, LOW);
      }
```

```
      //Left
      if (komen.equals("L")== true)
      {
      digitalWrite(22, LOW);
      digitalWrite(23, HIGH);
      digitalWrite(24, LOW);
      digitalWrite(25, HIGH);
      }
      //Stop
      else if (komen.equals("S")== true)
      {
      digitalWrite(22, LOW);
      digitalWrite(23, LOW);
      digitalWrite(24, LOW);
      digitalWrite(25, LOW);
      }

   komen="";
  }
```

4.2.2.1 Penjelasan source code pada Arduino IDE

a. Inisialisasi Terima Data

Inisialisasi terima data berfungsi untuk menampung dan membaca data yang dikirimkan oleh GUI dalam bentuk karakter. Pada inisialisasi terima data ini fungsi yang dipakai adalah fungsi *string* dan *char*. Fungsi *string* digunakan untuk mengidentifikasi variabel yang sudah di deklarisakan pada inisialisasi data yang terdapat pada *source code* GUI, sehingga fungsi *string* sering disebut juga dengan "*array of char*". Sedangkan fungsi *char* digunakan untuk menampung data bertipe tunggal. Gambar 4.28 menunjukkan program untuk inisialisasi terima data.

```
//Initial Receivier Data
char karakter;
String komen;
```

Gambar 4.28. Inisialisasi terima data

b. Pengaturan atau `void setup()`

Pengaturan atau fungsi `void setup()` berguna untuk menginisialisasi variabel, *mode pin*, memulai menggunakan *library*, memulai komunikasi serial, dll. Fungsi pengaturan ini hanya akan berjalan sekali, yaitu setiap *powerup* atau *restart board* Arduino.

Pada tahap pengaturan ini langkah pertama adalah membuka dan mengatur nilai *baud rate* komunikasi serial dengan menuliskan perintah `Serial.begin(57600)`. Untuk dapat berkomunikasi, nilai *baud rate* pada Arduino Mega 2560 dan laptop/PC harus sama, yaitu 57600.

Langkah kedua adalah konfigurasi *port* pada Arduino Mega 2560 dengan menuliskan perintah `pinMode(22, OUTPUT), pinMode(23, OUTPUT), pinMode(24, OUTPUT), pinMode(25, OUTPUT).` Dari perintah di atas *port* 22, *port* 23, *port* 24 dan *port* 25 pada Arduino di konfigurasikan sebagai *output*. *Output* inilah yang nantinya bertugas mengatur arah putar motor DC.

c. Eksekusi Perintah Program atau `void loop()`

Eksekusi Perintah Program atau `void loop()` berfungsi untuk mengeksekusi perintah program yang telah dibuat. Fungsi ini akan secara aktif mengontrol *board* Arduino baik membaca *input* ataupun merubah *output*.

Pada tahap ini langkah pertama adalah menuliskan perintah `while (Serial.available()>0)` fungsinya membaca *port* serial untuk mendapatkan jumlah karakter (*byte*) yang tersedia. Setelah menuliskan perintah tersebut langkah selanjutnya adalah menuliskan perintah `karakter=Serial.read();komen.concat(kara`

kter);delay(10); ke dalam fungsi tersebut. tujuannya agar karakter yang dikirimkan oleh GUI dapat dibaca dan eksekusi oleh Arduino.

Langkah kedua adalah membuat kondisi dimana data karakter yang dikirimkan oleh GUI disesuaikan dengan konfigurasi *output* untuk mengatur arah putar motor. Untuk mengatur robot bergerak maju sesuai navigasi yang ada pada GUI maka perlu pengkondisian karakter "F" sebagai *output*. Gambar 4.29 menunjukkan pengkondisian karakter "F".

```
//Forward
if (komen.equals("F") == true)
{
digitalWrite(22, HIGH);
digitalWrite(23, LOW);
digitalWrite(24, LOW);
digitalWrite(25, HIGH);
}
```

Gambar 4.29. Pengkondisian karakter "F"

Untuk mengatur robot bergerak mundur sesuai navigasi yang ada pada GUI maka perlu pengkondisian karakter "B" sebagai *output*. Gambar 4.30 menunjukkan pengkondisian karakter "B".

```
//Reverse
if (komen.equals("B")== true)
{
digitalWrite(22, LOW);
digitalWrite(23, HIGH);
digitalWrite(24, HIGH);
digitalWrite(25, LOW);
}
```

Gambar 4.30. Pengkondisian karakter "B"

Untuk mengatur robot berputar ke kanan sesuai navigasi yang ada pada GUI maka perlu pengkondisian

karakter “R” sebagai *output*. Gambar 4.31 menunjukkan pengkondisian karakter “R”.

```
//Right
if (komen.equals("R")== true)
{
digitalWrite(22, HIGH);
digitalWrite(23, LOW);
digitalWrite(24, HIGH);
digitalWrite(25, LOW);
}
```

Gambar 4.31. Pengkondisian karakter “R”

Untuk mengatur robot berputar ke kiri sesuai navigasi yang ada pada GUI maka perlu pengkondisian karakter “L” sebagai *output*. Gambar 4.32 menunjukkan pengkondisian karakter “L”.

```
//Left
if (komen.equals("L")== true)
{
digitalWrite(22, LOW);
digitalWrite(23, HIGH);
digitalWrite(24, LOW);
digitalWrite(25, HIGH);
}
```

Gambar 4.32. Pengkondisian karakter “L”

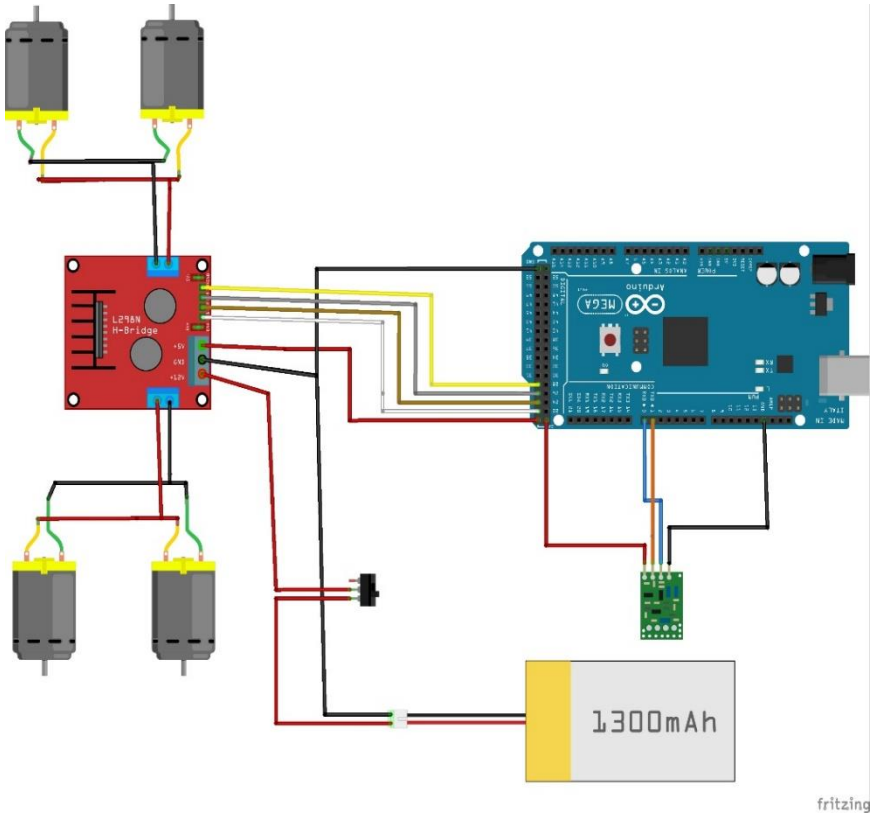
Untuk mengatur robot tidak bergerak sesuai navigasi yang ada pada GUI maka perlu pengkondisian karakter “S” sebagai *output*. Gambar 4.33 menunjukkan pengkondisian karakter “S”.

```
//Stop
else if (komen.equals("S")== true)
{
digitalWrite(22, LOW);
digitalWrite(23, LOW);
digitalWrite(24, LOW);
digitalWrite(25, LOW);
}
```

Gambar 4.33. Pengkondisian karakter “S”

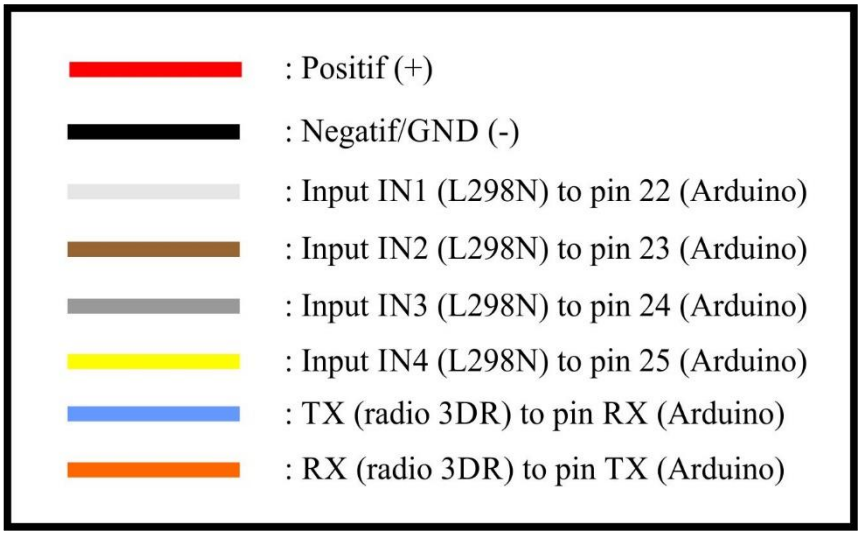
4.2.3 Perancangan dan *Wiring* Kit Robot Mobil Pintar 4WD

Dalam penelitian ini, perancangan dan *wiring* pada robot dibutuhkan guna mempermudah memahami pemasangan komponen – komponen yang dibutuhkan. Gambar 4.34 menunjukkan perancangan dan *wiring* komponen pada robot.



Gambar 4.34. Perancangan dan *wiring* komponen pada robot

Gambar 4.34. di atas menjeleskan rancangan dan *wiring* komponen robot pada penelitian ini. Pada *wiring* komponen robot di atas 2 buah motor DC paralel yang dihubungkan ke *output* A dan 2 buah motor DC paralel yang dihubungkan ke *output* B pada *driver* L298N *Dual H-Bridge*. Untuk menyuplai 4 buah motor tersebut, +12V *power* dan *power* GND pada *driver* L298N dihubungkan ke baterai *lithium polymer* (Li-Po) 3 cell. Untuk mengatur arah putar motor DC, *input driver* L298N dihubungkan ke pin Arduino Mega 2560. Konfigurasi *wiring input driver* L298N ke Arduino Mega 2560 adalah IN1 ke pin 22, IN2 ke pin 23, IN3 ke pin 24, IN4 ke pin 25. Untuk *power suply* Arduino Mega 2560, +5V power pada driver L298N dihubungkan ke +5V pada Arduino Mega 2560. Untuk modul radio 3DR, konfigurasinya adalah TX ke RX, RX ke TX, VCC ke +5V dan GND ke GND pada Arduino. Gambar 4.35 menjelaskan arti dari warna *wiring* pada komponen robot.



Gambar 4.35. Penjelasan warna *wiring* pada komponen robot

4.3 Cara Kerja Alat

Cara kerja alat pada penelitian ini mempunyai beberapa tahap, diantaranya :

1. Ketika *button* koneksi di tekan, maka seluruh kelas pada kelas *button* koneksi ini akan di eksekusi. Jika koneksi serial berhasil maka *text* "*Press to Connection*" pada GUI akan berubah menjadi "*Connected*" berwarna merah dan menampilkan *bar wifi* penuh. Jika koneksi serial tidak berhasil, maka kelas *error* akan di eksekusi dan akan muncul notifikasi jendala "*ERROR*", sehingga GUI akan tertutup secara otomatis. Agar koneksi berhasil, *baud rate* pada GUI dan Arduino harus sama dan *port* COM pada GUI harus sama dengan *port* COM pada laptop.
2. Setalah GUI dan robot sudah terkoneksi, maka masuk pada *button* navigasi. pada *button* navigasi ini *event* yang diberikan pada *button* maju, mundur, kanan dan kiri mempunyai *event* yang sama yaitu *ActionPerformed* dan *mousePressed*. Ketika *button* di tekan maka *event* yang bertugas adalah *event mousePressed*. Pada kelas *button* navigasi dengan *event mousePressed* akan mengeksekusi perintah `kirimData(FORWARD)` untuk *button* maju dengan mengirmkan karakter "F", `kirimData(REVERSE)` untuk *button* mundur dengan mengirimkan karakter "B", `kirimData(RIGHT)` untuk *button* kanan dengan mengirimkan karakter "R", dan `kirimData(LEFT)` untuk *button* kiri dengan mengirimkan karakter "L". Seluruh karakter di atas akan dikirimkan ke Arduino, kemudian Arduino akan mengeksekusi karakter yang di terima sesuai dengan pengkondisian yang sudah dibuat pada *source code* Arduino. Ketika *button* dilepas pada kelas *button* navigasi, baik itu *button* maju, mundur, kanan, atau kiri dengan *event actionPerformed* akan mengeksekusi perintah `kirimData(STOP)` dengan mengirimkan karakter "S". Karakter "S" ini akan di eksekusi oleh Arduino sesuai dengan pengkondisian yang sudah ada pada *source code* Arduino.

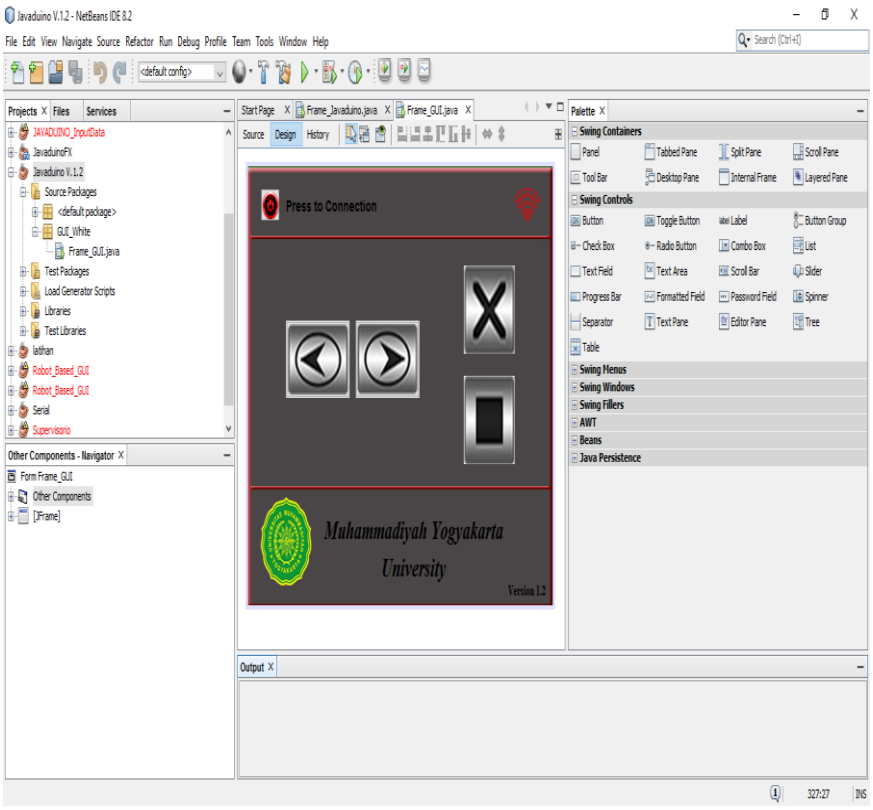
3. Untuk memutus koneksi serial GUI dengan Arduino, caranya dengan menekan *button* pemutus koneksi. Seluruh kelas atau *method* pada kelas ini akan di eksekusi dan *text* "*Connected*" berwarna merah pada GUI akan berubah menjadi "*Disconnected*" berwarna hitam serta *icon bar wifi* akan berubah menjadi kosong yang di ikuti dengan notifikasi jendela "Koneksi terputus".

4.4 Pengujian

Pengujian dilakukan untuk mengetahui apakah GUI bekerja sesuai dengan tujuan yang diharapkan oleh penulis pada penelitian ini.

4.4.1 Prinsip Kerja GUI

GUI di rancang dengan menggunakan *java application* dengan memanfaatkan *JFrame Form* yang terdapat pada *software* NetBeans 8.2. Di dalam *JFrame Form* terdapat jendela komponen yang digunakan untuk merancang sebuah GUI. Komponen yang digunakan untuk merancang GUI pada penelitian ini adalah 6 buah *button*, 7 buah *label* dan 8 buah gambar berformat *PNG*. Ukuran GUI pada penelitian ini 480 x 400 *pixels*. Untuk memasukkan komponen ke dalam *JFrame* caranya dengan *drag* dan *drop* komponen yang digunakan ke dalam *JFrame* sesuai dengan posisi yang di inginkan. Untuk mengganti *background, font, foreground, icon, text, border,* dan ukuran caranya dengan klik kanan komponen pilih *properties*. Setelah GUI selesai di rancang selanjutnya memasukkan *source code*. Untuk menjalankan GUI yang sudah di rancang caranya dengan kik tombol *clean* and *build* untuk meng-*compile source code*, jika notifikasi pesan "*build successful*" klik tombol *run* untuk menjalankan program. Gambar 4.36 menunjukkan rancangan tampilan GUI.



Gambar 4.36. Rancangan tampilan GUI

4.4.2 Prinsip Kerja Tombol Koneksi

Tombol koneksi digunakan untuk menghubungkan GUI dengan Arduino melalui komunikasi serial dengan memanfaatkan modul radio 3DR Telemetry Kit 433MHz sebagai medianya. Pada *source code* tombol koneksi mempunyai nama variabel `butwifi`. Di dalam kelas `butwifi` terdapat kelas `inisialKoneksi();` dan beberapa *method* diantaranya `butwifi.setVisible(false); butoff.setVisible(true); txtwifi.setIcon(new ImageIcon("src/baron.png")); lblwifi.setText("Connected"); lblwifi.setForeground(Color.red); inisialKoneksi();`.

Agar tombol memberikan aksi pada saat di tekan maka pada tombol koneksi di berikan *event actionPerformed*. Pada saat tombol

koneksi di tekan seluruh kelas dan method pada kelas tombol koneksi ini akan di eksekusi.

Pada kelas `inisialKoneksi();` merupakan tempat serial komunikasi dapat di akses dan kelas ini yang berfungsi sebagai penghubung *port* serial *COM3* dengan *baud rate* 57600.

Pada *method* `butwifi.setVisible(false);` menyatakan bahwa kelas `butwifi` tidak dapat dilihat ketika koneksi telah terhubung, artinya tombol koneksi akan hilang ketika koneksi telah terhubung. Ukuran tombol koneksi ini adalah 27 x 27 *pixels*.

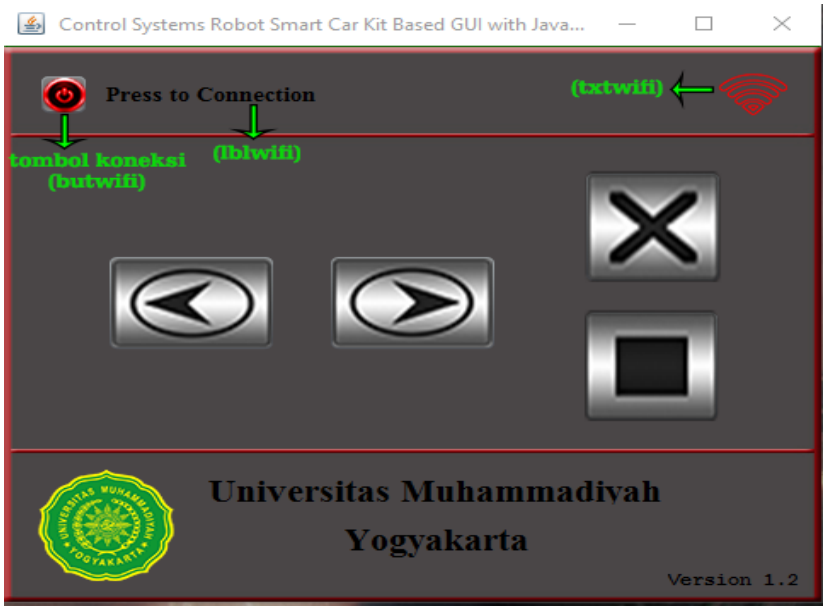
Pada *method* `butoff.setVisible(true);` menyatakan kelas `butoff` dapat dilihat ketika koneksi telah terhubung, artinya tombol pemutus koneksi akan tampil ketika koneksi telah terhubung. Ukuran tombol pemutus koneksi ini adalah 25 x 25 *pixels*.

Pada *method* `txtwifi.setIcon (new ImageIcon ("src/baron.png"));` menyatakan bahwa *label* dengan nama variabel `txtwifi` akan mengambil *icon* gambar baru dari folder src dengan nama gambar baron berformat PNG, artinya ketika koneksti telah terhubung *icon* gambar bar wifi kosong akan berubah menjadi icon gambar bar wifi penuh. Ukuran *icon* gambar ini adalah 40 x 40 *pixels*.

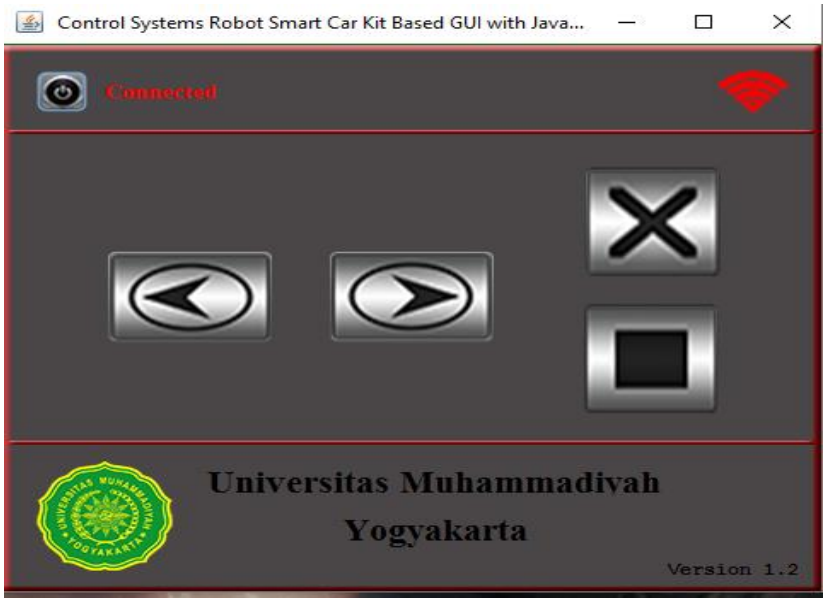
Pada *method* `lblwifi.setText("Connected");` menyatakan bahwa *label* dengan nama variabel `lblwifi` akan merubah *text* yang pada dasarnya "*Press to Connection*" menjadi *text* "*Connected*" ketika koneksi telah terhubung. Ukuran *text* dan *font* pada *label* ini adalah *vani, bold, 14 pt*.

Pada *method* `lblwifi.setForeground (Color.red);` menyatakan bahwa warna *text* pada *label* ini, yang pada dasarnya hitam akan berubah menjadi warna merah ketika

koneksi telah terhubung. Gambar 4.37 menunjukkan tampilan GUI sebelum tombol koneksi ditekan dan gambar 4.38 menunjukkan tampilan GUI setelah tombol koneksi ditekan.



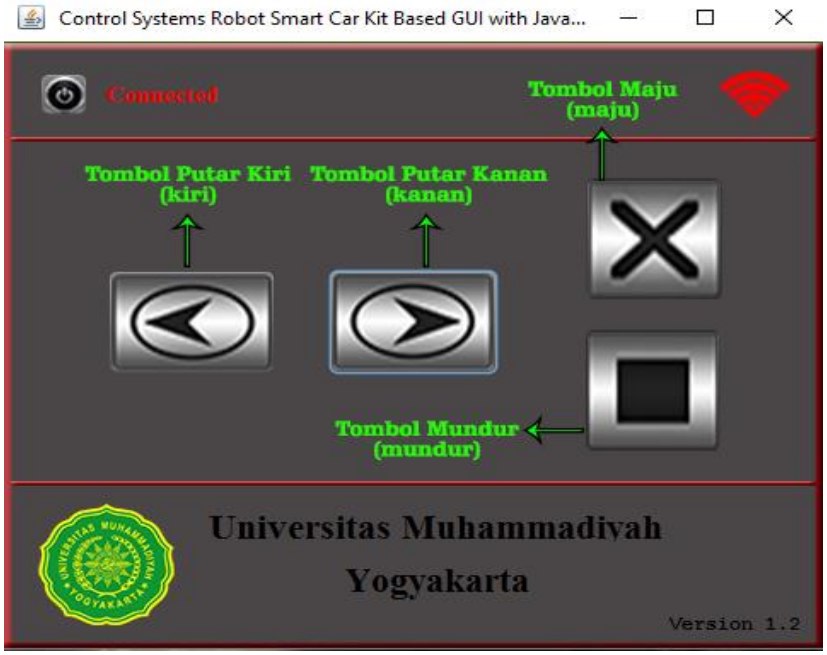
Gambar 4.37. Tampilan GUI sebelum tombol koneksi ditekan



Gambar 4.38. Tampilan GUI setelah tombol koneksi ditekan

4.4.3 Prinsip Kerja Tombol Navigasi

Tombol navigasi berfungsi untuk mengontrol robot baik itu maju, mundur, putar kiri, putar kanan, dan berhenti. Dalam pembuatan tombol navigasi menggunakan 4 buah tombol. Pada tiap tombol, nama variabel diganti dengan nama yang sesuai dengan arah gerak robot. Tujuannya untuk mempermudah dalam pengkodingan. Gambar 4.39 menunjukkan nama variabel untuk tombol navigasi.

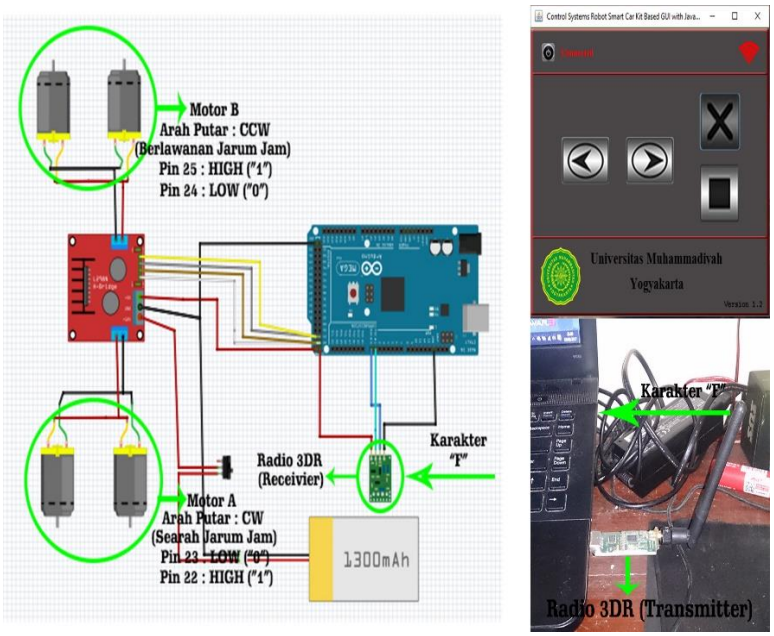


Gambar 4.39. Nama variabel untuk tombol navigasi

a. Tombol Maju

Tombol maju berfungsi untuk mengontrol robot agar bergerak maju. Pada kelas tombol maju dengan variabel `maju` diberi *event mousePressed*. *Event mousePressed* digunakan ketika *user* menekan *mouse*. Pada kelas `majuMousePressed` terdapat *method* `kirimData(FORWARD);` dan *method* `maju.setIcon (new ImageIcon("src/majuon.png"));`. kedua *method* ini akan di eksekusi ketika tombol maju ditekan dengan *mouse*. *Method* `kirimData(FORWARD);`

digunakan untuk mengirim inisialisasi data karaktek “F” melalui *serial port* pada laptop ke Arduino dengan media radio 3DR Telemetry Kit. Untuk mengontrol robot bergerak maju, pada *source code* Arduino karakter “F” diberi logika pin 22 = *HIGH* (“1”), pin 23 = *LOW* (“0”), pin 24 = *LOW* (“0”), pin 25 = *HIGH* (“1”). *Method* `maju.setIcon(new ImageIcon("src/majuon .png"));` digunakan untuk menampilkan *icon* gambar baru dengan nama file majuon yang terdapat pada folder src. Ukuran tombol maju ini adalah 133 x 109 *pixels*. Gambar 4.40 menunjukkan prinsip kerja tombol maju.

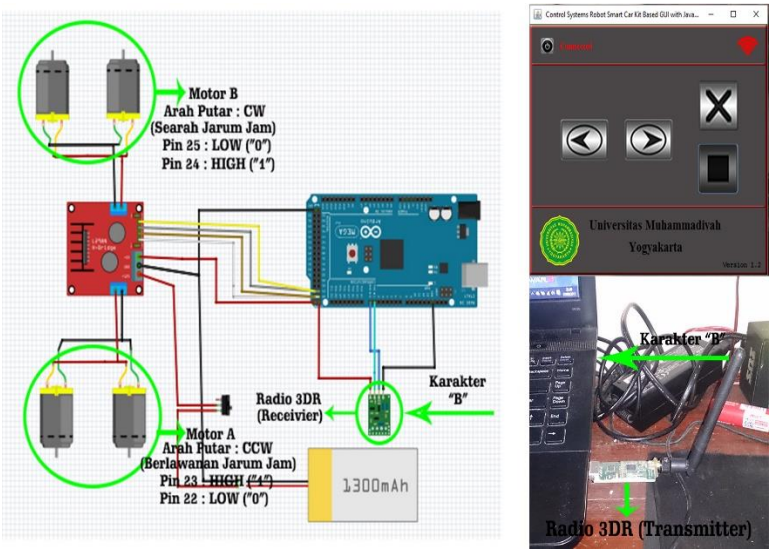


Gambar 4.40. Prinsip kerja tombol maju

b. Tombol Mundur

Tombol mundur berfungsi untuk mengontrol robot agar bergerak mundur. Pada kelas tombol mundur dengan variabel `mundur` diberi *event mousePressed*. *Event mousePressed* digunakan ketika *user* menekan *mouse*. Pada

kelas `mundurMousePressed` terdapat *method* `kirimData(REVERSE);` dan method `mundur.setIcon(new ImageIcon("src/munduron.png"));` , kedua *method* ini akan di eksekusi ketika tombol mundur ditekan dengan *mouse*. *Method* `kirimData(REVERSE);` digunakan untuk mengirim inisialisasi data karaktek "B" melalui *serial port* pada laptop ke Arduino dengan media radio 3DR Telemetry Kit. Untuk mengontrol robot bergerak mundur, pada *source code* Arduino karakter "B" diberi logika pin 22 = *LOW* ("0"), pin 23 = *HIGH* ("1"), pin 24 = *HIGH* ("1"), pin 25 = *LOW* ("0"). *Method* `mundur.setIcon(new ImageIcon("src/mundur on.png"));` digunakan untuk menampilkan *icon* gambar baru dengan nama file munduron yang terdapat pada folder src. Ukuran tombol mundur ini adalah 133 x 109 *pixels*. Gambar 4.41 menunjukkan prinsip kerja tombol mundur.

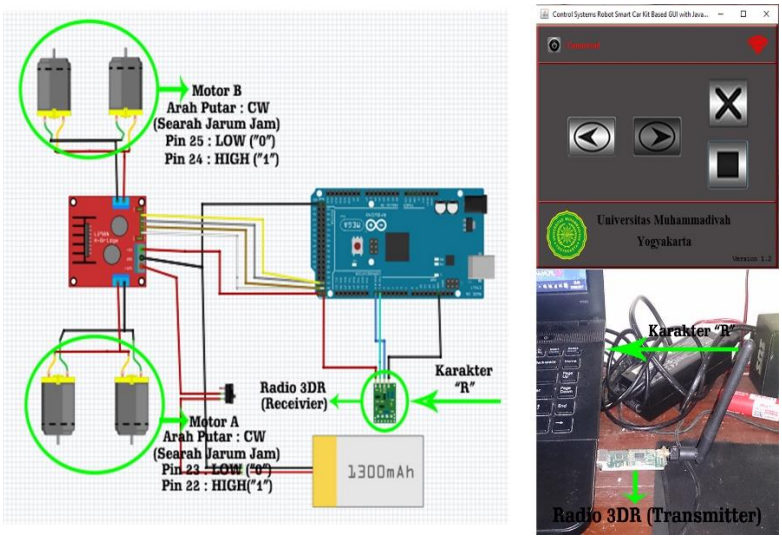


Gambar 4.41. Prinsip kerja tombol mundur

c. Tombol Putar Kanan

Tombol putar kanan berfungsi untuk mengontrol robot agar berputar ke kanan. Pada kelas tombol putar kanan

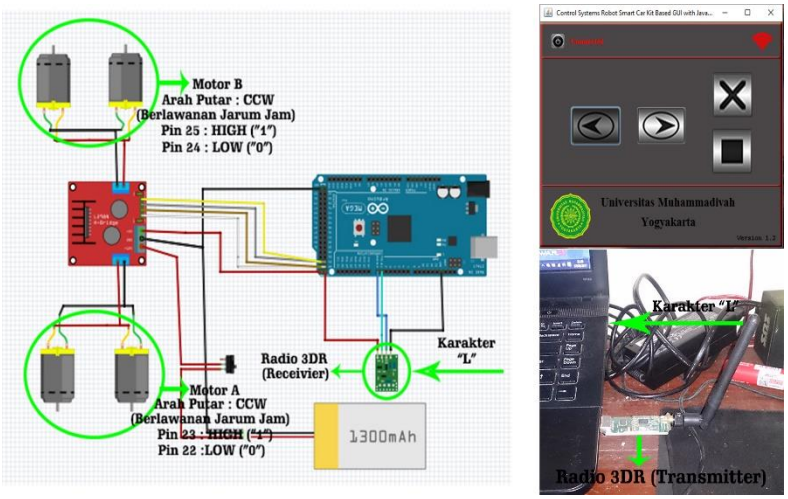
dengan variabel `kanan` diberi *event mousePressed*. *Event mousePressed* digunakan ketika *user* menekan *mouse*. Pada kelas `kananMousePressed` terdapat *method* `kirim Data(RIGHT);` dan *method* `kanan.setIcon(new ImageIcon("src/kananon.png"));` kedua *method* ini akan di eksekusi ketika tombol putar kanan ditekan dengan *mouse*. *Method* `kirimData(RIGHT);` digunakan untuk mengirim inisialisasi data karaktek "R" melalui *serial port* pada laptop ke Arduino dengan media radio 3DR Telemetry Kit. Untuk mengontrol robot berputar ke kanan, pada *source code* Arduino karakter "R" diberi logika pin 22 = *HIGH* ("1"), pin 23 = *LOW* ("0"), pin 24 = *HIGH* ("1"), pin 25 = *LOW* ("0"). *Method* `kanan.setIcon(new ImageIcon("src/kananon.png"));` digunakan untuk menampilkan *icon* gambar baru dengan nama file kananon yang terdapat pada folder src. Ukuran tombol putar kanan ini adalah 153 x 89 *pixels*. Gambar 4.42 menunjukkan prinsip kerja tombol putar kanan.



Gambar 4.42. Prinsip kerja tombol putar kanan

d. Tombol Putar Kiri

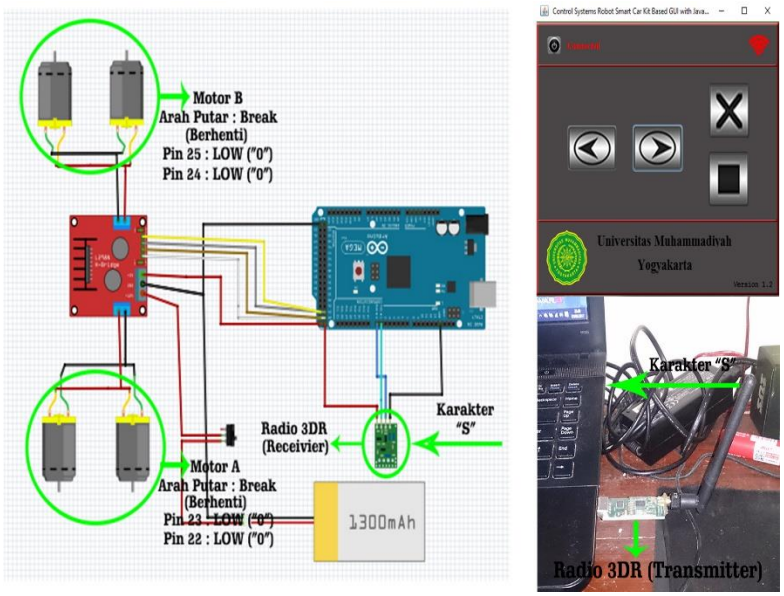
Tombol putar kiri berfungsi untuk mengontrol robot agar berputar ke kiri. Pada kelas tombol putar kiri dengan variabel `kanan` diberi *event mousePressed*. *Event mousePressed* digunakan ketika *user* menekan *mouse*. Pada kelas `kiriMousePressed` terdapat *method* `kirim Data(LEFT);` dan *method* `kiri.setIcon(new ImageIcon("src/kirion.png"));`, kedua *method* ini akan di eksekusi ketika tombol putar kiri ditekan dengan *mouse*. *Method* `kirimData(LEFT);` digunakan untuk mengirim inisialisasi data karaktek "L" melalui *serial port* pada laptop ke Arduino dengan media radio 3DR Telemetry Kit. Untuk mengontrol robot berputar ke kiri, pada *source code* Arduino karakter "L" diberi logika pin 22 = *LOW* ("0"), pin 23 = *HIGH* ("1"), pin 24 = *LOW* ("0"), pin 25 = *HIGH* ("1"). *Method* `kiri.setIcon(new ImageIcon ("src/kirion.png"));` digunakan untuk menampilkan *icon* gambar baru dengan nama file kirion yang terdapat pada folder src. Ukuran tombol putar kiri ini adalah 153 x 89 *pixels*. Gambar 4.43 menunjukkan prinsip kerja tombol putar kiri.



Gambar 4.43. Prinsip kerja tombol putar kiri

e. Tombol Berhenti

Tombol berhenti berfungsi untuk mengontrol robot agar berhenti bergerak. Tombol berhenti diwakili oleh seluruh kelas tombol navigasi yang diberi *event actionPerformed*. *Event actionPerformed* digunakan ketika *event action* terjadi. Pada kelas tombol berhenti ini terdapat *method* `kirimData(STOP);` dan *method* yang berfungsi untuk menampilkan gambar dari tiap tombol navigasi ke bentuk awal. kedua method ini akan di eksekusi ketika seluruh tombol navigasi tidak ditekan atau salah satu tombol navigasi yang sudah ditekan dilepas. *Method* `kirimData(STOP);` digunakan untuk mengirim inisialisasi data karaktek "S" melalui *serial port* pada laptop ke Arduino dengan media radio 3DR Telemetry Kit. Untuk mengontrol robot berhenti bergerak, pada *source code* Arduino karakter "S" diberi logika pin 22 = *LOW* ("0"), pin 23 = *LOW* ("0"), pin 24 = *LOW* ("0"), pin 25 = *LOW* ("0"). Gambar 4.44 menunjukkan prinsip kerja tombol berhenti.



Gambar 4.44. Prinsip kerja tombol berhenti

4.4.4 Prinsip Kerja Tombol Pemutus Koneksi

Tombol pemutus koneksi digunakan untuk memutuskan koneksi GUI dengan Arduino melalui komunikasi serial dengan memanfaatkan modul radio 3DR Telemetry Kit 433MHz sebagai medianya. Pada *source code* tombol pemutus koneksi mempunyai nama variabel `butoff`. Di dalam kelas `butoff` terdapat kelas `closeSerial();` dan beberapa *method* diantaranya `butwifi.setVisible(true);butoff.SetVisible(false);lblwifi.setText("Disconnected");lblwifi.setForeground(Color.black);txtwifi.setIcon(new ImageIcon("src/wifi-empty-xxl.png"));`. Agar tombol memberikan aksi pada saat di tekan maka pada tombol pemutus koneksi di berikan *event actionPerformed*. Pada saat tombol pemutus koneksi di tekan seluruh kelas dan *method* pada kelas tombol koneksi ini akan di eksekusi.

Pada kelas `closeSerial();` merupakan suatu kelas yang bertugas memutuskan koneksi serial atau dapat disebut juga dengan menutup *port* serial antara GUI dengan Arduino dan suatu kelas yang bertugas menampilkan jendela pesan "Koneksi terputus".

Pada *method* `butwifi.setVisible(true);` menyatakan bahwa kelas `butwifi` dapat dilihat ketika koneksi telah terputus, artinya tombol koneksi akan tampil ketika koneksi telah terputus. Ukuran tombol koneksi ini adalah 27 x 27 *pixels*.

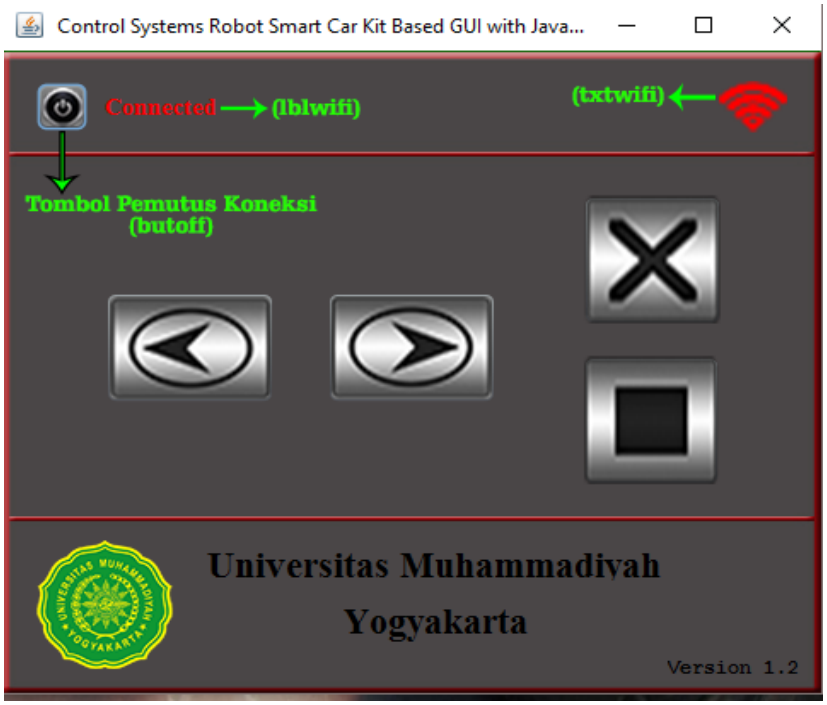
Pada *method* `butoff.setVisible(false);` menyatakan kelas `butoff` tidak dapat dilihat ketika koneksi telah terputus, artinya tombol pemutus koneksi akan hilang ketika koneksi telah terputus. Ukuran tombol pemutus koneksi ini adalah 25 x 25 *pixels*.

Pada *method* `lblwifi.setText("Disconnected");` menyatakan bahwa *label* dengan nama variabel `lblwifi` akan

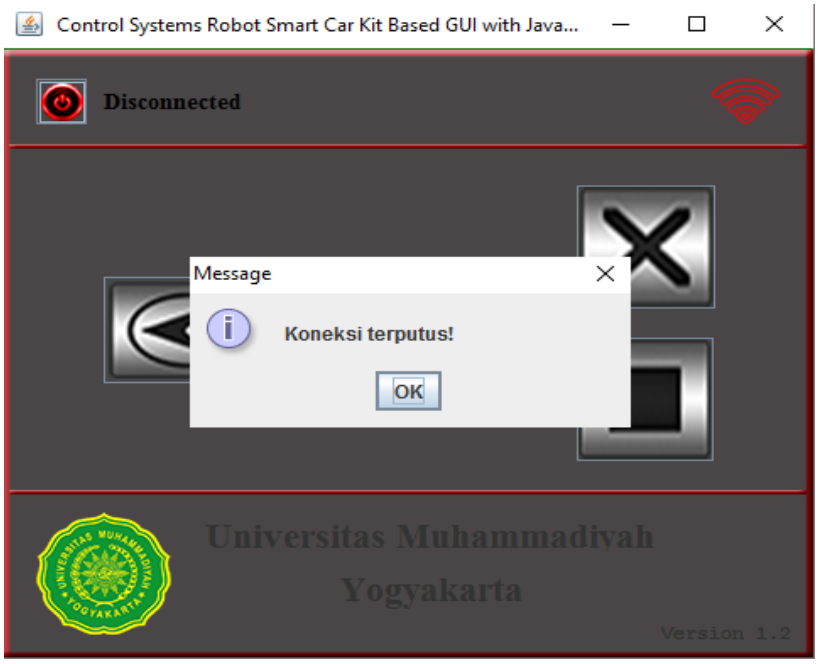
merubah *text* pada saat koneksi telah terputus menjadi "*Disconnected*". Ukuran *text* dan *font* pada *label* ini adalah *vani, bold, 14 pt*.

Pada *method* `lblwifi.setForeground(Color.black);` menyatakan bahwa warna *text* pada *label* ini akan berubah menjadi warna hitam ketika koneksi telah terputus.

Pada *method* `txtwifi.setIcon(new ImageIcon("src/wifi-empty-xxl.png"));` menyatakan bahwa *label* dengan nama variabel `txtwifi` akan mengambil *icon* gambar baru dari folder src dengan nama gambar wifi-empty-xxl berformat PNG, artinya ketika koneksti telah terputus *icon* gambar *bar wifi* penuh akan berubah menjadi *icon* gambar *bar wifi* kosong. Ukuran *icon* gambar ini adalah 40 x 40 *pixels*. Gambar 4.45 menunjukkan tampilan GUI pada saat koneksi telah terhubung dan gambar 4.46 menunjukkan tampilan GUI setelah tombol pemutus koneksi ditekan.



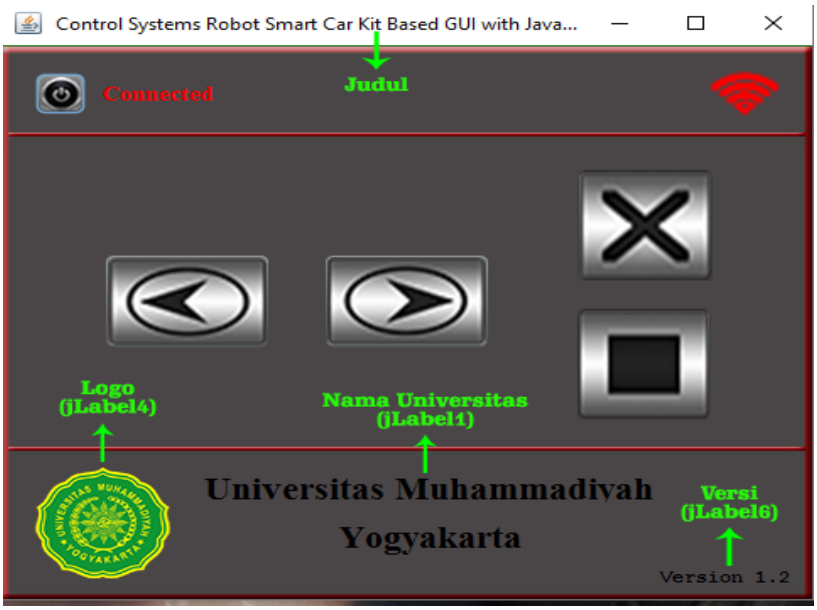
Gambar 4.45. Tampilan GUI pada saat koneksi terhubung



Gambar 4.46. Tombol pemutus koneksi ditekan

4.4.5 Judul , Nama Universitas, Logo dan Versi

Pada GUI terdapat judul, nama universitas, logo, dan versi. Judul GUI dibuat dengan menuliskan *method* `setTitle("Control Systems Robot Smart Car Kit Based GUI with Java Programming. ");` pada kelas `Frame_GUI`. Nama universitas dibuat dengan menuliskan *teks* "Universitas Muhammdiyah Yogyakarta" pada variabel `jLabel1` dengan *font Times New Roman, 22, Bold*. Logo dibuat dengan mengganti *icon* gambar pada variabel `jLabel4` dengan logo Universitas Muhammdiyah Yogyakarta. Versi dibuat dengan menuliskan *teks* "*Version* 1.2" pada variabel `jLabel6` dengan *font Courier New, 12, Bold*. Gambar 4.47 menunjukkan tampilan judul, nama universitas, logo dan versi.



Gambar 4.47. Tampilan judul, nama universitas, logo dan versi

4.4.6 Jarak jangkauan Transmisi

Pengukuran jarak jangkuan transmisi antara GUI dengan robot dilakukan di Stadion Maguwoharjo. Jarak yang dapat dijangkau sekitar 960 meter.

Setelah pengujian dilakukan dan GUI berjalan dengan baik, selanjutnya merubah GUI menjadi sebuah *software* yang dapat berjalan di atas sistem operasi *Windows* dan *Macintosh*. Agar dapat berjalan di atas sistem operasi *Windows* dan *Macintosh*, GUI dibuat mejadi sebuah *installer* menggunakan *software* Advanced Installer 13.5. *Software* ini bernama JavaDuino V1.2.