LAMPIRAN B: KODE SUMBER C++ UNTUK PERANGKAT LUNAK ROBOT.

Seluruh file ini dapat di akses di URL:

<https://github.com/mekatronik-achmadi/tugas_akhir/tree/master/software/qrobotvision3>

Seluruh kode sumber ini dibangun dengan operating system ubuntu atau debian. Kode sumber terdiri dari file:

1. qrobot.pro
2. qrobot.cpp
3. qrobot.h
4. qrobot.ui
5. main.cpp
6. SerialPort/

Untuk dapat mengkompilasi kode sumber ini maka dibutuhkan paket berikut:

1. libopencv-dev
2. libqt4-dev
3. build-essential

Karena berbasis qt4 maka untuk mengkompilasi dapat menggunakan program qmake dan make. Untuk memanggilnya dibutuhkan privilage root agar dapat mengakses /dev/tty\*

Untuk file di dalam folder SerialPort adalah file pustaka QextSerialPort yang digunakan tanpa ada perubahan apapun.

1. File qrobot.pro

#-------------------------------------------------

#

# Project created by QtCreator 2014-10-21T09:39:07

#

#-------------------------------------------------

QT +**=** core gui

greaterThan(QT\_MAJOR\_VERSION, 4)**:** QT += widgets

TARGET **=** qrobot

TEMPLATE **=** app

SOURCES +**=** main.cpp\

qrobot.cpp \

SerialPortLibs/posix\_qextserialport.cpp \

SerialPortLibs/qextserialbase.cpp \

SerialPortLibs/qextserialenumerator.cpp \

SerialPortLibs/qextserialport.cpp

HEADERS +**=** qrobot.h \

SerialPortLibs/posix\_qextserialport.h \

SerialPortLibs/qextserialbase.h \

SerialPortLibs/qextserialenumerator.h \

SerialPortLibs/qextserialport.h

FORMS +**=** qrobot.ui

DEPENDPATH +**=** .

INCLUDEPATH +**=** .

INCLUDEPATH +**=** /usr/local/include/opencv2

LIBS +**=** -lopencv\_core -lopencv\_highgui -lopencv\_imgproc

DEFINES +**=** \_TTY\_POSIX\_

1. File qrobot.cpp

#include "qrobot.h"

#include "ui\_qrobot.h"

int iLowH **=** 100**;**

int iHighH **=** 130**;**

int iLowS **=** 87**;**

int iHighS **=** 183**;**

int iLowV **=** 80**;**

int iHighV **=** 182**;**

uint minArea **=** 31**;**

uint maxArea **=** 22500**;**

qrobot**::**qrobot**(**QWidget **\***parent**)** **:**

QMainWindow**(**parent**),**my\_port**(**0**),**

ui**(new** Ui**::**qrobot**)**

**{**

ui**->**setupUi**(this);**

my\_cam\_timer **=** **new** QTimer**(this);**

QObject**::**connect**(**my\_cam\_timer**,**SIGNAL**(**timeout**()),this,**SLOT**(**img\_proc**()));**

my\_cam\_timer**->**start**(**10**);**

cam**.**open**(**0**);**

**if** **(** **!**cam**.**isOpened**()){**

std**::**cout **<<** "Cannot open the web cam" **<<** std**::**endl**;**

**}**

cv**::**namedWindow**(**"Control"**,** CV\_WINDOW\_AUTOSIZE**);**

cv**::**createTrackbar**(**"LowH"**,** "Control"**,** **&**iLowH**,** 179**);**

cv**::**createTrackbar**(**"HighH"**,** "Control"**,** **&**iHighH**,** 179**);**

cv**::**createTrackbar**(**"LowS"**,** "Control"**,** **&**iLowS**,** 255**);**

cv**::**createTrackbar**(**"HighS"**,** "Control"**,** **&**iHighS**,** 255**);**

cv**::**createTrackbar**(**"LowV"**,** "Control"**,** **&**iLowV**,** 255**);**

cv**::**createTrackbar**(**"HighV"**,** "Control"**,** **&**iHighV**,** 255**);**

cam**.**set**(**CV\_CAP\_PROP\_FRAME\_WIDTH**,** 320**);**

cam**.**set**(**CV\_CAP\_PROP\_FRAME\_HEIGHT**,** 240**);**

**this->**setWindowTitle**(**"QRobot [Closed]"**);**

ui**->**actionOpen**->**setEnabled**(true);**

ui**->**actionClose**->**setEnabled**(false);**

ui**->**actionStop**->**setEnabled**(false);**

ui**->**actionFoward**->**setEnabled**(false);**

ui**->**actionBackward**->**setEnabled**(false);**

ui**->**actionRight**->**setEnabled**(false);**

ui**->**actionLeft**->**setEnabled**(false);**

qrobot**::**on\_actionOpen\_triggered**();**

my\_timer **=** **new** QTimer**(this);**

QObject**::**connect**(**my\_timer**,**SIGNAL**(**timeout**()),this,**SLOT**(**booted**()));**

my\_timer**->**start**(**500**);**

my\_move **=** **new** QTimer**(this);**

QObject**::**connect**(**my\_move**,**SIGNAL**(**timeout**()),this,**SLOT**(**move\_proc**()));**

my\_move**->**start**(**500**);**

QObject**::**connect**(**my\_port**,**SIGNAL**(**readyRead**()),this,**SLOT**(**next\_move**()));**

**}**

qrobot**::~**qrobot**()**

**{**

**delete** ui**;**

**}**

void qrobot**::**on\_actionOpen\_triggered**()**

**{**

**if(**my\_port**){**

**if(**my\_port**->**isOpen**()){**

my\_port**->**close**();**

**}**

**delete** my\_port**;**

my\_port**=NULL;**

**}**

QString my\_device**=** "/dev/ttyUSB0"**;**

BaudRateType my\_baud **=** BAUD38400**;**

my\_port **=** **new** QextSerialPort**(**my\_device**,**QextSerialPort**::**Polling**);**

my\_port**->**setBaudRate**(**my\_baud**);**

my\_port**->**setDataBits**(**DATA\_8**);**

my\_port**->**setParity**(**PAR\_NONE**);**

my\_port**->**setStopBits**(**STOP\_1**);**

my\_port**->**setFlowControl**(**FLOW\_OFF**);**

my\_port**->**setTimeout**(**100**);**

**if(!**my\_port**->**open**(**QIODevice**::**ReadWrite**)){**

**delete** my\_port**;**

my\_port**=NULL;**

QString s**=**"Cannot open device at "**;**

s **+=** my\_device**;**

QMessageBox**::**critical**(this,**"Error"**,**s**);**

**this->**setWindowTitle**(**"QRobot [Closed]"**);**

ui**->**actionOpen**->**setEnabled**(true);**

ui**->**actionClose**->**setEnabled**(false);**

ui**->**actionStop**->**setEnabled**(false);**

ui**->**actionFoward**->**setEnabled**(false);**

ui**->**actionBackward**->**setEnabled**(false);**

ui**->**actionRight**->**setEnabled**(false);**

ui**->**actionLeft**->**setEnabled**(false);**

**}**

**else{**

**this->**setWindowTitle**(**"QRobot [Opened]"**);**

ui**->**actionOpen**->**setEnabled**(false);**

ui**->**actionClose**->**setEnabled**(true);**

ui**->**actionStop**->**setEnabled**(true);**

ui**->**actionFoward**->**setEnabled**(true);**

ui**->**actionBackward**->**setEnabled**(true);**

ui**->**actionRight**->**setEnabled**(true);**

ui**->**actionLeft**->**setEnabled**(true);**

**}**

**}**

void qrobot**::**on\_actionClose\_triggered**()**

**{**

**if(**my\_port**){**

**if(**my\_port**->**isOpen**()){**

my\_port**->**close**();**

**}**

**delete** my\_port**;**

my\_port**=NULL;**

**this->**setWindowTitle**(**"QRobot [Closed]"**);**

ui**->**actionOpen**->**setEnabled**(true);**

ui**->**actionClose**->**setEnabled**(false);**

ui**->**actionStop**->**setEnabled**(false);**

ui**->**actionFoward**->**setEnabled**(false);**

ui**->**actionBackward**->**setEnabled**(false);**

ui**->**actionRight**->**setEnabled**(false);**

ui**->**actionLeft**->**setEnabled**(false);**

**}**

**}**

void qrobot**::**on\_actionFoward\_triggered**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"foward\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**on\_actionBackward\_triggered**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"backward\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**on\_actionRight\_triggered**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"right\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**on\_actionLeft\_triggered**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"left\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**on\_actionStop\_triggered**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"stop\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**booted**()**

**{**

my\_timer**->**stop**();**

QByteArray comdata**=**"booted\n"**;**

my\_port**->**write**(**comdata**);**

my\_timer**->**start**(**500**);**

**}**

void qrobot**::**img\_proc**()**

**{**

cv**::**Mat imgOriginal**;**

bool bSuccess **=** cam**.**read**(**imgOriginal**);**

**if** **(!**bSuccess**)** **{**

std**::**cout **<<** "Cannot read a frame from video stream" **<<** std**::**endl**;**

**}**

cv**::**Mat imgsqr**=** cv**::**Mat**::**zeros**(** imgOriginal**.**size**(),** CV\_8UC3 **);**

cv**::**Mat imgHSV**;**

cv**::**cvtColor**(**imgOriginal**,** imgHSV**,** cv**::**COLOR\_BGR2HSV**);**

cv**::**Mat imgThresholded**;**

cv**::**inRange**(**imgHSV**,** cv**::**Scalar**(**iLowH**,** iLowS**,** iLowV**),** cv**::**Scalar**(**iHighH**,** iHighS**,** iHighV**),** imgThresholded**);**

cv**::**imshow**(**"Thresholded Image"**,** imgThresholded**);**

imgRows **=** imgThresholded**.**rows**;**

imgCols **=** imgThresholded**.**cols**;**

imgBiner **=** cv**::**Mat**(**imgRows**,**imgCols**,**CV\_8U**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgRows**;**i**++){**

**for(**j**=**0**;**j**<**imgCols**;**j**++){**

**if(**imgThresholded**.**at**<**uchar**>(**i**,**j**)==**255**){**imgBiner**.**at**<**uchar**>(**i**,**j**)=**1**;}**

**else{**imgBiner**.**at**<**uchar**>(**i**,**j**)=**0**;}**

**}**

**}**

xval**=**cv**::**Mat**(**1**,**imgCols**,**CV\_16U**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgCols**;**i**++){**

xval**.**at**<**ushort**>(**0**,**i**)=**0**;**

**for(**j**=**0**;**j**<**imgRows**;**j**++){**

xval**.**at**<**ushort**>(**0**,**i**)=**xval**.**at**<**ushort**>(**0**,**i**)+**imgBiner**.**at**<**uchar**>(**j**,**i**);**

**}**

**}**

mArea**=**cv**::**Mat**(**1**,**1**,**CV\_16U**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgCols**;**i**++){**

mArea**.**at**<**ushort**>(**0**,**0**)=**mArea**.**at**<**ushort**>(**0**,**0**)+**xval**.**at**<**ushort**>(**0**,**i**);**

**}**

Area**=**mArea**.**at**<**ushort**>(**0**,**0**);**

**if((**Area**>**minArea**)&&(**Area**<=**maxArea**)){**

mxSum**=**cv**::**Mat**(**1**,**1**,**CV\_32F**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgCols**;**i**++){**

mxSum**.**at**<**float**>(**0**,**0**)=**mxSum**.**at**<**float**>(**0**,**0**)+(**i**\***xval**.**at**<**ushort**>(**0**,**i**));**

**}**

xSum**=**mxSum**.**at**<**float**>(**0**,**0**);**

xcen**=**xSum**/**Area**;**

yval**=**cv**::**Mat**(**1**,**imgRows**,**CV\_16U**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgRows**;**i**++){**

yval**.**at**<**ushort**>(**0**,**i**)=**0**;**

**for(**j**=**0**;**j**<**imgCols**;**j**++){**

yval**.**at**<**ushort**>(**0**,**i**)=**yval**.**at**<**ushort**>(**0**,**i**)+**imgBiner**.**at**<**uchar**>(**i**,**j**);**

**}**

**}**

mySum**=**cv**::**Mat**(**1**,**1**,**CV\_32F**,**cv**::**Scalar**(**0**));**

**for(**i**=**0**;**i**<**imgRows**;**i**++){**

mySum**.**at**<**float**>(**0**,**0**)=**mySum**.**at**<**float**>(**0**,**0**)+(**i**\***yval**.**at**<**ushort**>(**0**,**i**));**

**}**

ySum**=**mySum**.**at**<**float**>(**0**,**0**);**

ycen**=**ySum**/**Area**;**

**}**

imgCen **=** cv**::**Mat**(** imgOriginal**.**size**(),** CV\_8UC3**,**cv**::**Scalar**(**0**,**0**,**0**));**

vCen**=**25**;**

cv**::**circle**(**imgCen**,**cv**::**Point**(**xcen**,**ycen**),**vCen**,**cv**::**Scalar**(**0**,**255**,**0**),**2**);**

imgRad**=** cv**::**Mat**(** imgOriginal**.**size**(),** CV\_8UC3**,**cv**::**Scalar**(**0**,**0**,**0**));**

vRad**=**45**;**

cv**::**circle**(**imgRad**,**cv**::**Point**(**imgCols**/**2**,**imgRows**/**2**),**vRad**,**cv**::**Scalar**(**0**,**255**,**0**),**2**);**

cv**::**imshow**(**"Thresholded Image"**,** imgThresholded**);**

imgFinal **=** imgOriginal **+** imgRad**+**imgCen**;**

cv**::**imshow**(**"Final"**,** imgFinal**);**

**}**

void qrobot**::**move\_proc**(){**

xPos**=** xcen**;**

yPos**=** xcen**;**

xDef**=** imgCols**/**2**;**

yDef**=** imgRows**/**2**;**

vDef **=** 25**;**

vArea**=** Area**;**

**if((**Area**>**minArea**)&&(**Area**<=**maxArea**)){**

std**::**cout **<<** "xPos= " **<<** xPos **<<** " " **<<** "yPos= " **<<** yPos **<<** " " **<<** "Area= " **<<** vArea **<<** std**::**endl**;**

**if(**xPos**<**xDef**){**

**if((**xDef**-**xPos**)>**vDef**){**

qrobot**::**on\_actionLeft\_triggered**();**

std**::**cout **<<** "LEFT" **<<** std**::**endl**;**

**}**

**else{**

qrobot**::**on\_actionFoward\_triggered**();**

std**::**cout **<<** "FOWARD" **<<** std**::**endl**;**

**}**

**}**

**else** **if(**xPos**>**xDef**){**

**if((**xPos**-**xDef**)>**vDef**){**

qrobot**::**on\_actionRight\_triggered**();**

std**::**cout **<<** "RIGHT" **<<** std**::**endl**;**

**}**

**else{**

qrobot**::**on\_actionFoward\_triggered**();**

std**::**cout **<<** "FOWARD" **<<** std**::**endl**;**

**}**

**}**

**else** **if(**xPos**==**xDef**){**

qrobot**::**on\_actionFoward\_triggered**();**

std**::**cout **<<** "FOWARD" **<<** std**::**endl**;**

**}**

**}**

**else{**

std**::**cout **<<** "xPos= " **<<** xPos **<<** " " **<<** "yPos= " **<<** yPos **<<** " " **<<** "Area= " **<<** vArea **<<** std**::**endl**;**

std**::**cout **<<** "STOP" **<<** std**::**endl**;**

**}**

**}**

1. File qrobot.h

#ifndef QROBOT\_H

#define QROBOT\_H

#include <QMainWindow>

#include <QMessageBox>

#include <QTimer>

#include "SerialPortLibs/qextserialport.h"

#include "SerialPortLibs/qextserialenumerator.h"

#include <iostream>

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

**namespace** Ui **{**

class qrobot**;**

**}**

class qrobot **:** public QMainWindow

**{**

Q\_OBJECT

public**:**

explicit qrobot**(**QWidget **\***parent **=** 0**);**

**~**qrobot**();**

private slots**:**

void on\_actionOpen\_triggered**();**

void on\_actionClose\_triggered**();**

void on\_actionFoward\_triggered**();**

void on\_actionBackward\_triggered**();**

void on\_actionRight\_triggered**();**

void on\_actionLeft\_triggered**();**

void on\_actionStop\_triggered**();**

void booted**();**

void img\_proc**();**

void move\_proc**();**

private**:**

Ui**::**qrobot **\***ui**;**

QextSerialPort **\***my\_port**;**

QTimer**\*** my\_cam\_timer**;**

QTimer**\*** my\_timer**;**

cv**::**VideoCapture cam**;**

uint imgRows**;**

uint imgCols**;**

uint i**,**j**;**

cv**::**Mat imgBiner**;**

uint sqr**;**

cv**::**Mat mArea**;**

uint Area**;**

cv**::**Mat xval**;**

cv**::**Mat mxSum**;**

float xSum**;**

uint xcen**;**

cv**::**Mat yval**;**

cv**::**Mat mySum**;**

float ySum**;**

uint ycen**;**

cv**::**Mat imgRad**;**

uint vRad**;**

cv**::**Mat imgCen**;**

uint vCen**;**

cv**::**Mat imgFinal**;**

QTimer**\*** my\_move**;**

uint xPos**,**yPos**;**

uint xDef**,**yDef**;**

uint vDif**,**vDef**;**

uint vArea**;**

**};**

#endif // QROBOT\_H

1. File qrobot.ui

<?xml version=**"1.0"** encoding=**"UTF-8"**?>

<ui version=**"4.0"**>

<class>**qrobot**</class>

<widget class=**"QMainWindow"** name=**"qrobot"**>

<property name=**"geometry"**>

<rect>

<x>**0**</x>

<y>**0**</y>

<width>**442**</width>

<height>**88**</height>

</rect>

</property>

<property name=**"windowTitle"**>

<string>**qrobot**</string>

</property>

<widget class=**"QWidget"** name=**"centralWidget"**/>

<widget class=**"QMenuBar"** name=**"menuBar"**>

<property name=**"geometry"**>

<rect>

<x>**0**</x>

<y>**0**</y>

<width>**442**</width>

<height>**20**</height>

</rect>

</property>

</widget>

<widget class=**"QToolBar"** name=**"mainToolBar"**>

<attribute name=**"toolBarArea"**>

<enum>**TopToolBarArea**</enum>

</attribute>

<attribute name=**"toolBarBreak"**>

<bool>**false**</bool>

</attribute>

<addaction name=**"actionOpen"**/>

<addaction name=**"actionClose"**/>

<addaction name=**"actionFoward"**/>

<addaction name=**"actionBackward"**/>

<addaction name=**"actionRight"**/>

<addaction name=**"actionLeft"**/>

<addaction name=**"actionStop"**/>

</widget>

<widget class=**"QStatusBar"** name=**"statusBar"**/>

<action name=**"actionOpen"**>

<property name=**"text"**>

<string>**Open**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+O**</string>

</property>

</action>

<action name=**"actionClose"**>

<property name=**"text"**>

<string>**Close**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+C**</string>

</property>

</action>

<action name=**"actionFoward"**>

<property name=**"text"**>

<string>**Foward**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+W**</string>

</property>

</action>

<action name=**"actionBackward"**>

<property name=**"text"**>

<string>**Backward**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+S**</string>

</property>

</action>

<action name=**"actionRight"**>

<property name=**"text"**>

<string>**Right**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+D**</string>

</property>

</action>

<action name=**"actionLeft"**>

<property name=**"text"**>

<string>**Left**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+A**</string>

</property>

</action>

<action name=**"actionStop"**>

<property name=**"text"**>

<string>**Stop**</string>

</property>

<property name=**"shortcut"**>

<string>**Ctrl+X**</string>

</property>

</action>

</widget>

<layoutdefault spacing=**"6"** margin=**"11"**/>

<resources/>

<connections/>

</ui>

File main.cpp

#include "qrobot.h"

#include <QApplication>

int main**(**int argc**,** char **\***argv**[]){**

QApplication a**(**argc**,** argv**);**

qrobot w**;**

w**.**show**();**

**return** a**.**exec**();**

**}**