LAMPIRAN

a) Source Code Keseluruhan Bola Warna Kuning

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import time
import numpy as np
import imutils

defaultSpeed = 50
windowCenter = 320
centerBuffer = 10
pwmBound = float(50)
cameraBound = float(320)
kp = pwmBound / cameraBound
leftBound = int(windowCenter - centerBuffer)
rightBound = int(windowCenter + centerBuffer)
error = 0
ballPixel = 0

#GPIO
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#Pin definitions
rightFwd = 7
```

```
rightRev = 11
leftFwd = 13
leftRev = 15

#GPIO initialization
GPIO.setup(leftFwd, GPIO.OUT)
GPIO.setup(leftRev, GPIO.OUT)
GPIO.setup(rightFwd, GPIO.OUT)
GPIO.setup(rightRev, GPIO.OUT)

#Disable movement at startup
GPIO.output(leftFwd, False)
GPIO.output(leftRev, False)
GPIO.output(rightFwd, False)
GPIO.output(rightRev, False)

#PWM Initialization

rightMotorFwd = GPIO.PWM(rightFwd, 50)
leftMotorFwd = GPIO.PWM(leftFwd, 50)
rightMotorRev = GPIO.PWM(rightRev, 50)
leftMotorRev = GPIO.PWM(leftRev, 50)
rightMotorFwd.start(defaultSpeed)
leftMotorFwd.start(defaultSpeed)
leftMotorRev.start(defaultSpeed)
rightMotorRev.start(defaultSpeed)
def updatePwm(rightPwm, leftPwm):
    rightMotorFwd.ChangeDutyCycle(rightPwm)
```

```
    leftMotorFwd.ChangeDutyCycle(leftPwm)

def pwmStop():
    rightMotorFwd.ChangeDutyCycle(0)
    rightMotorRev.ChangeDutyCycle(0)
    leftMotorFwd.ChangeDutyCycle(0)
    leftMotorRev.ChangeDutyCycle(0)

#Camera setup
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 15
rawCapture = PiRGBArray(camera, size = (640, 480))

time.sleep(0.1)

#disatukan
lower_yellow = np.array([10, 65, 90])
upper_yellow = np.array([40, 255, 245])

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

    image = frame.array
    output = image.copy()
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
```

```
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    output = cv2.bitwise_and(output, output, mask=mask)
    #output = cv2.dilate(output, None, iterations=2)
    #output = cv2.erode(output, None, iterations=2)
    gray = cv2.cvtColor(output, cv2.COLOR_BGR2GRAY)
    #_, binary = cv2.threshold(gray, 1, 255, cv2. THRESH_BINARY)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 3, 500, minRadius
= 10, maxRadius = 200, param1 = 100,  param2 = 60)
    #cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
    ballPixel = 0

    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        for (x, y, radius) in circles:

            cv2.circle(output, (x, y), radius, (0, 255, 0), 4)
            #cv2.rectangle(output, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)

            if radius > 10:
                ballPixel = x
            else:
                ballPixel = 0

    cv2.imshow("output", output)
    cv2.imshow("original", image)
    #cv2.imshow("original", mask)
    cv2.namedWindow("window")
```

```
key = cv2.waitKey(1) & 0xFF
rawCapture.truncate(0)

#Proportional controller
if ballPixel == 0:
   #print "no ball"
   error = 0
   pwmStop()
elif (ballPixel < leftBound) or (ballPixel > rightBound):
   error = windowCenter - ballPixel
   pwmOut = abs(error * kp)
   #print ballPixel
   turnPwm = pwmOut + defaultSpeed
   if  ballPixel < (leftBound):
      #print "left side"
      if radius > 50 and ballPixel < 110:
         print (ballPixel)
         updatePwm(defaultSpeed, 20)
      else:
         updatePwm(turnPwm, defaultSpeed)
   elif ballPixel > (rightBound):
      #print "right side"
      if radius > 50 and ballPixel > 540:
         print (ballPixel)
         updatePwm(20, defaultSpeed)
      else:
         updatePwm(defaultSpeed, turnPwm)
else:
```

```
        #print "middle"
        if (radius < 40):
            updatePwm(defaultSpeed, defaultSpeed)
        else:
            pwmStop()

    if key == ord('q'):
        break

cv2.destroyAllWindows()
camera.close()
pwmStop()
GPIO.cleanup()
```

b) Source Code Keseluruhan Bola Warna Merah

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import time
import numpy as np
import imutils

defaultSpeed = 50
windowCenter = 320
centerBuffer = 10
pwmBound = float(50)
cameraBound = float(320)
kp = pwmBound / cameraBound
```

```
leftBound = int(windowCenter - centerBuffer)
rightBound = int(windowCenter + centerBuffer)
error = 0
ballPixel = 0

#GPIO
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#Pin definitions
rightFwd = 7
rightRev = 11
leftFwd = 13
leftRev = 15

#GPIO initialization
GPIO.setup(leftFwd, GPIO.OUT)
GPIO.setup(leftRev, GPIO.OUT)
GPIO.setup(rightFwd, GPIO.OUT)
GPIO.setup(rightRev, GPIO.OUT)

#Disable movement at startup
GPIO.output(leftFwd, False)
GPIO.output(leftRev, False)
GPIO.output(rightFwd, False)
GPIO.output(rightRev, False)

#PWM Initialization
```

```
rightMotorFwd = GPIO.PWM(rightFwd, 50)
leftMotorFwd = GPIO.PWM(leftFwd, 50)
rightMotorRev = GPIO.PWM(rightRev, 50)
leftMotorRev = GPIO.PWM(leftRev, 50)
rightMotorFwd.start(defaultSpeed)
leftMotorFwd.start(defaultSpeed)
leftMotorRev.start(defaultSpeed)
rightMotorRev.start(defaultSpeed)
def updatePwm(rightPwm, leftPwm):
    rightMotorFwd.ChangeDutyCycle(rightPwm)
    leftMotorFwd.ChangeDutyCycle(leftPwm)

def pwmStop():
    rightMotorFwd.ChangeDutyCycle(0)
    rightMotorRev.ChangeDutyCycle(0)
    leftMotorFwd.ChangeDutyCycle(0)
    leftMotorRev.ChangeDutyCycle(0)

#Camera setup
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 15
rawCapture = PiRGBArray(camera, size = (640, 480))

time.sleep(0.1)

#disatukan
```

```
lower_yellow = np.array([168,175,0])
upper_yellow = np.array([255,255,255])

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

    image = frame.array
    output = image.copy()
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    output = cv2.bitwise_and(output, output, mask=mask)
    #output = cv2.dilate(output, None, iterations=2)
    #output = cv2.erode(output, None, iterations=2)
    gray = cv2.cvtColor(output, cv2.COLOR_BGR2GRAY)
    #_, binary = cv2.threshold(gray, 1, 255, cv2. THRESH_BINARY)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 3, 500, minRadius
= 10, maxRadius = 200, param1 = 100,  param2 = 60)
    #cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
    ballPixel = 0

    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        for (x, y, radius) in circles:

            cv2.circle(output, (x, y), radius, (0, 255, 0), 4)
```

```
        #cv2.rectangle(output, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)

        if radius > 10:
            ballPixel = x
        else:
            ballPixel = 0

    cv2.imshow("output", output)
    cv2.imshow("original", image)
    #cv2.imshow("original", mask)
    cv2.namedWindow("window")
    key = cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)

    #Proportional controller
    if ballPixel == 0:
        #print "no ball"
        error = 0
        pwmStop()
    elif (ballPixel < leftBound) or (ballPixel > rightBound):
        error = windowCenter - ballPixel
        pwmOut = abs(error * kp)
        #print ballPixel
        turnPwm = pwmOut + defaultSpeed
        if  ballPixel < (leftBound):
            #print "left side"
            if radius > 50 and ballPixel < 110:
                print (ballPixel)
```

```
            updatePwm(defaultSpeed, 20)
        else:
            updatePwm(turnPwm, defaultSpeed)
    elif ballPixel > (rightBound):
        #print "right side"
        if radius > 50 and ballPixel > 540:
            print (ballPixel)
            updatePwm(20, defaultSpeed)
        else:
            updatePwm(defaultSpeed, turnPwm)
  else:
    #print "middle"
    if (radius < 40):
        updatePwm(defaultSpeed, defaultSpeed)
    else:
        pwmStop()

  if key == ord('q'):
    break

cv2.destroyAllWindows()
camera.close()
pwmStop()
GPIO.cleanup()
```

b) Source Code Keseluruhan Bola Warna Merah

```
from picamera.array import PiRGBArray
from picamera import PiCamera
```

```
import cv2
import time
import numpy as np
import imutils

defaultSpeed = 50
windowCenter = 320
centerBuffer = 10
pwmBound = float(50)
cameraBound = float(320)
kp = pwmBound / cameraBound
leftBound = int(windowCenter - centerBuffer)
rightBound = int(windowCenter + centerBuffer)
error = 0
ballPixel = 0

#GPIO
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#Pin definitions
rightFwd = 7
rightRev = 11
leftFwd = 13
leftRev = 15

#GPIO initialization
GPIO.setup(leftFwd, GPIO.OUT)
```

```
GPIO.setup(leftRev, GPIO.OUT)
GPIO.setup(rightFwd, GPIO.OUT)
GPIO.setup(rightRev, GPIO.OUT)

#Disable movement at startup
GPIO.output(leftFwd, False)
GPIO.output(leftRev, False)
GPIO.output(rightFwd, False)
GPIO.output(rightRev, False)

#PWM Initialization

rightMotorFwd = GPIO.PWM(rightFwd, 50)
leftMotorFwd = GPIO.PWM(leftFwd, 50)
rightMotorRev = GPIO.PWM(rightRev, 50)
leftMotorRev = GPIO.PWM(leftRev, 50)
rightMotorFwd.start(defaultSpeed)
leftMotorFwd.start(defaultSpeed)
leftMotorRev.start(defaultSpeed)
rightMotorRev.start(defaultSpeed)
def updatePwm(rightPwm, leftPwm):
    rightMotorFwd.ChangeDutyCycle(rightPwm)
    leftMotorFwd.ChangeDutyCycle(leftPwm)

def pwmStop():
    rightMotorFwd.ChangeDutyCycle(0)
    rightMotorRev.ChangeDutyCycle(0)
    leftMotorFwd.ChangeDutyCycle(0)
```

```
    leftMotorRev.ChangeDutyCycle(0)

#Camera setup
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 15
rawCapture = PiRGBArray(camera, size = (640, 480))

time.sleep(0.1)

#disatukan
lower_yellow = np.array([100,100,60])
upper_yellow = np.array([123,255,255])

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

    image = frame.array
    output = image.copy()
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    output = cv2.bitwise_and(output, output, mask=mask)
    #output = cv2.dilate(output, None, iterations=2)
    #output = cv2.erode(output, None, iterations=2)
    gray = cv2.cvtColor(output, cv2.COLOR_BGR2GRAY)
```

```
    #_, binary = cv2.threshold(gray, 1, 255, cv2. THRESH_BINARY)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 3, 500, minRadius
= 10, maxRadius = 200, param1 = 100,  param2 = 60)
    #cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
    ballPixel = 0

    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        for (x, y, radius) in circles:

            cv2.circle(output, (x, y), radius, (0, 255, 0), 4)
            #cv2.rectangle(output, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)

            if radius > 10:
                ballPixel = x
            else:
                ballPixel = 0

    cv2.imshow("output", output)
    cv2.imshow("original", image)
    #cv2.imshow("original", mask)
    cv2.namedWindow("window")
    key = cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)

    #Proportional controller
    if ballPixel == 0:
        #print "no ball"
```

```
        error = 0
        pwmStop()
    elif (ballPixel < leftBound) or (ballPixel > rightBound):
        error = windowCenter - ballPixel
        pwmOut = abs(error * kp)
        #print ballPixel
        turnPwm = pwmOut + defaultSpeed
        if  ballPixel < (leftBound):
            #print "left side"
            if radius > 50 and ballPixel < 110:
                print (ballPixel)
                updatePwm(defaultSpeed, 20)
            else:
                updatePwm(turnPwm, defaultSpeed)
        elif ballPixel > (rightBound):
            #print "right side"
            if radius > 50 and ballPixel > 540:
                print (ballPixel)
                updatePwm(20, defaultSpeed)
            else:
                updatePwm(defaultSpeed, turnPwm)
    else:
        #print "middle"
        if (radius < 40):
            updatePwm(defaultSpeed, defaultSpeed)
        else:
            pwmStop()
```

```
    if key == ord('q'):
        break

cv2.destroyAllWindows()
camera.close()
pwmStop()
GPIO.cleanup()
```