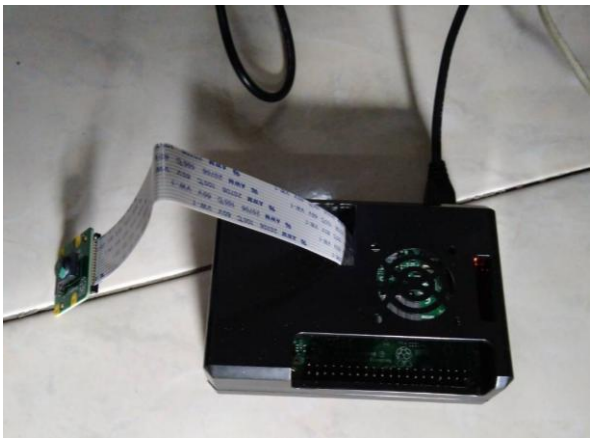
III.3. Realisasi

Selanjutnya adalah tahap dalam merealisasikan rancangan dan simulasi yang telah dibuat agar dapat menjadi sistem sesungguhnya.

III.3.1 Realisasi Perangkat Keras



Gambar III.4. Realisasi Perangkat keras

Pada gambar III.4. masih dalam bentuk Raspberry Pi yang terhubung dengan Pi Camera V2. Yang menjadi inti dari perangkat keras yang mendukung pengolah citra. Dimana mikrokontroler ini akan diprogram sedemikian rupa yang nanti akan dibahas di bagian realisasi perangkat lunak.

III.3.2 Realisasi Perangkat Lunak

Pada sistem ini direalisasikan pembuatan Perangkat Lunak berupa program dari Pengolah Citra dengan deteksi Canny untuk direalisasikan pada mikrokontroler.

1. Program Kamera untuk mengambil gambar

```
import cv2, time
# Membuat Objek, nol untuk Kamera Default
cap = cv2.VideoCapture(0)
a=0
i=0
while(True):
    a+=1
    # Membuat Objek Frame
    check, frame = cap.read()
    """
    print(check)
    print(frame) # Menampilkan Nilai Matrix Gambar
    """
    frameg = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Menampilan Gambar Video
    # Untuk Menggunakan 'C' sebagai alternatif Capture
    cv2.imshow("imshow", frame)
    key=cv2.waitKey(30)
    if key==ord('c'):
        i+=1
        cv2.imshow("imshow2",frame)
        cv2.imwrite(r'target_uang\Uang'+str(i)+'k.png', frame)
        print("Gambar telah Difoto")
    if key==ord('x'):
        break
# Mematikan Kamera
cap.release()
cv2.destroyAllWindows()
```

Program di atas, merupakan pemrograman python untuk menampilkan gambar visual dengan menggunakan *library OpenCV*. Selain menampilkan gambar visual, program di atas juga berfungsi mengambil gambar dengan menekan tombol ‘C’ pada keyboard komputer dan menutup program dengan menekan tombol ‘X’.

2. Program Memotong gambar ke bentuk ukuran uang / persegi Panjang

```
import cv2
import numpy as np
import math
from transfrom import transform

class ImageTransform  :
    def __init__(self, url):
        self.image = cv2.imread(url)
        self.shape = ""

    def resize(self, newWidth, interpolationMethod):
        interpolationAlgoritma={
            "area": cv2.INTER_AREA,
            "nearest": cv2.INTER_NEAREST,
            "linear":cv2.INTER_LINEAR,
            "cubic":cv2.INTER_CUBIC,
            "lanczos4": cv2.INTER_LANCZOS4,
        }
        image = self.image
        heigth, width, _ = image.shape
        imageRatio = newWidth / width
        newDimention = (newWidth, int(heigth * imageRatio))
        self.image = cv2.resize(image,
newDimention,interpolationAlgoritma[interpolationMethod] )
        return self

    def gausianBlur(self,matrixSize):
        greyImage =
cv2.cvtColor(self.image.copy(),cv2.COLOR_BGR2GRAY)
        self.greyImage =  cv2.GaussianBlur(greyImage, matrixSize, 0)
        return self

    def edgeDetect(self, method = 'canny'):
        self.gausianBlur((7,7))
        self.newImage = self.image.copy()

        edge = cv2.Canny(self.greyImage, 100,100)
        if method == 'laplace' : edge =
cv2.Laplacian(self.greyImage,cv2.CV_8UC1)
        contours, _ = cv2.findContours(edge.copy(),1,1)
```

```
        self.contours = contours
        maxArea = 0
        maxPeri = 0
        maxCountour = 0

        # for i in contours:
        #     area = cv2.contourArea(i)
        #     if area > maxArea:
        #         maxArea = area
        #         maxCountour = i
        for i in contours :
            peri = cv2.arcLength(i, True)
            if peri > maxPeri:
                maxPeri = peri
                maxCountour = i


        self.maxCountour = maxCountour
        return self

    def hougeLine (self):
        self.gausianBlur((7,7))
        newImage = self.image.copy()
        edge = cv2.Canny(self.greyImage, 100,100)
        lines = cv2.HoughLines(edge,1,np.pi / 180, 120, None, 0, 0)
        if lines is not None:
            for i in range(0, len(lines)):
                rho = lines[i][0][0]
                theta = lines[i][0][1]
                a = math.cos(theta)
                b = math.sin(theta)
                x0 = a * rho
                y0 = b * rho
                pt1 = (int(x0 + 1000*(-b)), int(y0 + 1000*(a)))
                pt2 = (int(x0 - 1000*(-b)), int(y0 - 1000*(a)))
                cv2.line(newImage, pt1, pt2, (0,0,255), 3, cv2.LINE_AA)
        cv2.imwrite('report/image9-hough.jpg',newImage)

    def applyContour(self):
        newImage = cv2.drawContours(self.image, self.contours,-
1,(0,0,255), 1)
        return newImage

    def getRectangle(self):
        rect= cv2.minAreaRect(self.maxCountour)
        box = cv2.boxPoints(rect)
        box = np.intc(box)
        peri=cv2.arcLength(box,True)
```

```
        approx=cv2.approxPolyDP(box,0.02*peri,True)
        w,h,arr = transform(approx)

        pts2=np.float32([[0,0],[w,0],[0,h],[w,h]])
        pts1=np.float32(arr)
        M=cv2.getPerspectiveTransform(pts1,pts2)
        newImage=cv2.warpPerspective(self.image, M,(w,h))
        ratio = newImage.size / self.image.size
        print(ratio)
        if ratio > 0.2 and ratio < 1: self.image = newImage
        h,w,_ = self.image.shape
        print(w,h)
        if w > h : self.rotate(90)
        return self
    def cropImage(self):
        contour = self.maxCountour
        peri=cv2.arcLength(contour,True)
        approx=cv2.approxPolyDP(contour,0.02*peri,True)
        print(approx)
        w,h,arr = transform(approx)
        pts2=np.float32([[0,0],[w,0],[0,h],[w,h]])
        pts1=np.float32(arr)
        M=cv2.getPerspectiveTransform(pts1,pts2)
        newImage=cv2.warpPerspective(self.image, M,(w,h))
        self.image = newImage
        return self
    def rotate(self, degre):
        heigth,width,_ = self.image.shape
        rotationMatrix = cv2.getRotationMatrix2D((width/2, heigth/2),
degre,1)
        cos = np.abs(rotationMatrix[0, 0])
        sin = np.abs(rotationMatrix[0, 1])
        nW = int((heigth * sin) + (width * cos))
        nH = int((heigth * cos) + (width * sin))
        rotationMatrix[0, 2] += (nW / 2) - width/2
        rotationMatrix[1, 2] += (nH / 2) - heigth/2
        image = cv2.warpAffine(self.image,rotationMatrix,(heigth,
width))
        self.image = image
        return self
    def write(self, name):
        cv2.imwrite(name, self.image)
        return self
    def show(self, name):
        cv2.imshow(name, self.image)
        cv2.waitKey(0)
        return self
img = ImageTransform('../target_uang/Uang100rb.jpg')
img.resize(600, 'area').edgeDetect().cropImage().show('output')
```

3. Program Komparasi hasil gambar yang diambil dengan referensi

```
import cv2
import numpy as np
import glob
import imutils

# Menyimpan data template //Langkah 1//
template_data=[]
FileRef=glob.glob('tmp_ref/*.jpg')
for myFile in FileRef:
    image = cv2.imread(myFile)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.Canny(image, 50, 200)
    template_data.append(image)
# Perulangan untuk template matching //Langkah 2//
for tmp in template_data:
    (tM, tW) = tmp.shape[:2]
    cv2.imshow("Template", tmp)
# Melakukan Looping untuk mengunggah data gambar dan melakukan
Langkah 2 //Langkah 3//
    for imageP in glob.glob('target_uang/image100-crop.jpg'):
        # Convert ke Abu/Grayscale
        imageS = cv2.imread(imageP)
        gray = cv2.cvtColor(imageS, cv2.COLOR_BGR2GRAY)
        cv2.imshow("Imageg", gray)
        found = None
        # Melakukan Pengulangan untuk Scaling Gambar //Langkah 4//
        for scale in np.linspace(0.2, 1.0, 20)[::-1]:
            # Merescale gambar sesuai dengan skala yang diberikan
//Langkah 5//
            resized = imutils.resize(gray, width = int(gray.shape[1] * scale))
            r = gray.shape[1] / float(resized.shape[1])

            # Jika Gambar hasil rescale < tmp_ref
            # Break Loop
            if resized.shape[0] < tM or resized.shape[1] < tW:
                break

            # Deteksi Edge pada Gambar yang telah di scale //Langkah 6//
            # Melakukan template Matching
            edged = cv2.Canny(resized, 50, 200)
            cv2.imshow("Images", edged)
            result = cv2.matchTemplate(edged, tmp,
cv2.TM_CCOEFF_NORMED)
            (_, maxVal, _, maxLoc) = cv2.minMaxLoc(result)
            # Jika bertemu dengan variabel dengan skala baru, maka simpan
```

```
//Langkah 7//
          if found is None or maxVal > found[0]:
             found = (maxVal, maxLoc, r)
             if maxVal >= 0.3:
                rep = "Uang"
                imageP1 = imageP
                imageP1 = imageP1.replace(rep,"")
                imageP1 = imageP1.replace("\\","")
                imageP1 = imageP1.replace(".jpg","")
                print("uang",imageP1,"Terdeteksi")
       # Mengeluarkan variabel gambar yg di telah di skala ke bentuk
aslinya //Langkah 8//
       # Berikan Kotak pada gambar asli jika cocok
       (maxVal, maxLoc, r) = found
       (startX, startY) = (int(maxLoc[0] * r), int(maxLoc[1] * r))
       (endX, endY) = (int((maxLoc[0] + tW) *r), int((maxLoc[1] + tW)
* r))
       if maxVal >= 0.4:
          cv2.rectangle(imageS, (startX, startY), (endX, endY), (0, 0,
255), 2)
          cv2.imshow("Image", imageS)
          cv2.waitKey(0)
```

Program di atas, merupakan pemrograman python untuk menampilkan gambar visual dengan menggunakan *library OpenCV*. Berbeda dengan program sebelumnya, program ini khusus untuk melakukan Komparasi dua buah gambar, satu gambar target dan satu gambar referensi. Teknik yang digunakan dalam komparasinya adalah dengan menggunakan *Coeff. Template Matching*. Dengan nilai kesamaan yang ditargetkan di atas 30%.