```
from gopigo import *
from picamera.array import PiRGBArray
from picamera import PiCamera
import sys
import pygame
import time
import os
import io
import picamera
import cv2
import numpy


servo_pos=90
face_detect='OFF'

pygame.init()
screen = pygame.display.set_mode((700, 700))
pygame.display.set_caption('Remote Control Window')

background = pygame.Surface(screen.get_size())
background = background.convert()
background.fill((250, 250, 250))

instructions = '''
                        BASIC GOPIGO CONTROL GUI

This is a basic example for the GoPiGo Robot control

(Be sure to put focus on thi window to control the gopigo!)

Press:
      ->w: Move GoPiGo Robot forward
      ->a: Turn GoPiGo Robot left
      ->d: Turn GoPiGo Robot right
      ->s: Move GoPiGo Robot backward
      ->t: Increase speed
      ->g: Decrease speed
      ->z: Exit


Servo CONTROLS
h: move servo left
k: move servo right
j: move servo home

PRESS "N" TO SCAN FOR FACES
''';
size_inc=22
index=0
for i in instructions.split('\n'):
    font = pygame.font.Font(None, 36)
    text = font.render(i, 1, (10, 10, 10))
    background.blit(text, (10,10+size_inc*index))
    index+=1

# Blit everything to the screen
screen.blit(background, (0, 0))
pygame.display.flip()

while True:
    event = pygame.event.wait();
    if (event.type == pygame.KEYUP):
        stop();
        continue;
    if (event.type != pygame.KEYDOWN):
        continue;
```

```
68      char = event.unicode;
69      if char=='w':
70          fwd()   ;# Move forward
71      elif char=='a':
72          left(); # Turn left
73      elif char=='d':
74          right();# Turn Right
75      elif char=='s':
76          bwd();# Move back
77      elif char=='t':
78          increase_speed();   # Increase speed
79      elif char=='g':
80          decrease_speed();   # Decrease speed
81      elif char=='z':
82          print "\nExiting";      # Exit
83          sys.exit();
84          elif char=='h':
85                  enable_servo()
86                  servo_pos=servo_pos+10
87                  if servo_pos > 180:
88                          servo_pos=180
89                  servo(servo_pos)
90                  time.sleep(.1)
91                  disable_servo()
92          elif char=='j':
93                  enable_servo()
94                  servo_pos=90
95                  servo(90)
96                  time.sleep(1)
97                  disable_servo()
98          elif char=='k':
99                  enable_servo()
100                 servo_pos=servo_pos-10
101                 if servo_pos < 0:
102                         servo_pos=0
103                 servo(servo_pos)
104                 time.sleep(.1)
105                 disable_servo()
106         elif char=='n':
107                 print "\nScanning"
108                 servo(180)
109                 stream = io.BytesIO()
110                 with picamera.PiCamera() as camera:
111                     camera.resolution = (2560, 1840)
112                     camera.capture(stream, format='jpeg')
113                     time.sleep(1)
114                 buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
115                 image = cv2.imdecode(buff, 1)
116                 face_cascade =
                    cv2.CascadeClassifier('/home/pi/Desktop/OpenCV/Cascades/faces.xml')
117                 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
118                 faces = face_cascade.detectMultiScale(gray, 1.1, 5)
119                 print "Found "+str(len(faces))+" faces"
120                 for (x,y,w,h) in faces:
121                     cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),10)
122                 cv2.imwrite('result1.jpg',image)
123
124                 servo(135)
125                 stream = io.BytesIO()
126                 with picamera.PiCamera() as camera:
127                     camera.resolution = (2560, 1840)
128                     camera.capture(stream, format='jpeg')
129                     time.sleep(1)
130                 buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
131                 image = cv2.imdecode(buff, 1)
132                 face_cascade =
                    cv2.CascadeClassifier('/home/pi/Desktop/OpenCV/Cascades/faces.xml')
```

```
133                 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
134                 faces = face_cascade.detectMultiScale(gray, 1.1, 5)
135                 print "Found "+str(len(faces))+" faces"
136                 for (x,y,w,h) in faces:
137                     cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),10)
138                 cv2.imwrite('result2.jpg',image)
139 
140                 servo(90)
141                 stream = io.BytesIO()
142                 with picamera.PiCamera() as camera:
143                     camera.resolution = (2560, 1840)
144                     camera.capture(stream, format='jpeg')
145                     time.sleep(1)
146                 buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
147                 image = cv2.imdecode(buff, 1)
148                 face_cascade =
                    cv2.CascadeClassifier('/home/pi/Desktop/OpenCV/Cascades/faces.xml')
149                 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
150                 faces = face_cascade.detectMultiScale(gray, 1.1, 5)
151                 print "Found "+str(len(faces))+" faces"
152                 for (x,y,w,h) in faces:
153                     cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),10)
154                 cv2.imwrite('result3.jpg',image)
155 
156                 servo(45)
157                 stream = io.BytesIO()
158                 with picamera.PiCamera() as camera:
159                     camera.resolution = (2560, 1840)
160                     camera.capture(stream, format='jpeg')
161                     time.sleep(1)
162                 buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
163                 image = cv2.imdecode(buff, 1)
164                 face_cascade =
                    cv2.CascadeClassifier('/home/pi/Desktop/OpenCV/Cascades/faces.xml')
165                 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
166                 faces = face_cascade.detectMultiScale(gray, 1.1, 5)
167                 print "Found "+str(len(faces))+" faces"
168                 for (x,y,w,h) in faces:
169                     cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),10)
170                 cv2.imwrite('result4.jpg',image)
171 
172                 servo(0)
173                 stream = io.BytesIO()
174                 with picamera.PiCamera() as camera:
175                     camera.resolution = (2560, 1840)
176                     camera.capture(stream, format='jpeg')
177                     time.sleep(1)
178                 buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
179                 image = cv2.imdecode(buff, 1)
180                 face_cascade =
                    cv2.CascadeClassifier('/home/pi/Desktop/OpenCV/Cascades/faces.xml')
181                 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
182                 faces = face_cascade.detectMultiScale(gray, 1.1, 5)
183                 print "Found "+str(len(faces))+" faces"
184                 for (x,y,w,h) in faces:
185                     cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),10)
186                 cv2.imwrite('result5.jpg',image)
187 
188                 servo(90)
189         elif char=='b':
190                 from picamera.array import PiRGBArray
191                 from picamera import PiCamera
192                 import time
193                 import cv2
194                 camera = PiCamera()
195                 camera.resolution = (640, 480)
196                 camera.framerate = 40
```

```
197             rawCapture = PiRGBArray(camera, size=(640, 480))
198             time.sleep(0.1)
199             for frame in camera.capture_continuous(rawCapture, format="bgr",
                use_video_port=True):
200                     image = frame.array
201                 cv2.imshow("Frame", image)
202                     key = cv2.waitKey(1) & 0xFF
203                     rawCapture.truncate(0)
204                 if key == ord("q"):
205                         break
206
207
```