ARM Cortex-A53

The ARM Cortex-A53 is one of the first two microarchitectures implementing the ARMv8-A 64-bit instruction set designed by ARM Holdings. The Cortex-A53 is a superscalar processor, capable of dual-issuing some instructions. It was announced October 30th, 2012 and is marketed by ARM as either a stand-alone, more energy-efficient alternative to the more powerful Cortex-A57 microarchitecture, or to be used alongside a more powerful microarchitecture in a big.LITTLE configuration. It is available as an IP core to licensees, commensurate with other ARM intellectual property and processor designs. The ARM Cortex-A53 processor has been used in the LeMaker HiKey since 2015 and Raspberry Pi 3 since February 2016.

Raspberry Pi

Raspberry Pi is a wonderful Development platform based on ARM microprocessor. With its high computational power, it can work out wonders in hands of electronics hobbyists or students. All this can be possible only if we know how to make it interact with the real world and analyse the data through some output device. There are many sensors which can detect certain parameters from the real time world and transfer it to a digital world and we analyse them viewing them either in a 7-segment display, LCD screen or some other display.

7 Segment Display

7 Segment Display has seven segments in it and each segment has one LED inside it to display the numbers by lighting up the corresponding segments i.e., if we want the 7-segment to display the number "5" then you need to glow segment a, f, g, c, and d by making their corresponding pins low. There are two types of 7-segment displays: Common Cathode and Common Anode, here we are using Common Anode seven segment display. The 7-segment display has 16 pins as shown below.

- 7 or 8 segment pins
- Ground pin
- 4-digit pins

Interfacing Raspberry Pi With 7 Segment Display

Sl. No	**Raspberry Pi GPIO number**	**Raspberry Pi PIN number**	**7-Segment name**	**7-Seg pin number (here in this module)**
1	GPIO 26	PIN 37	Segment a	1
2	GPIO 19	PIN 35	Segment b	2
3	GPIO 13	PIN 33	Segment c	3
4	GPIO 6	PIN 31	Segment d	4
5	GPIO 5	PIN 29	Segment e	5
6	GPIO 11	PIN 23	Segment f	6
7	GPIO 9	PIN 21	Segment g	7
8	GPIO 10	PIN 19	Segment DP	8
9	GPIO 7	PIN 26	Digit 1	13
10	GPIO 8	PIN 24	Digit 2	14
11	Ground	Ground	Ground	11

CODE

```
import RPi.GPIO as GPIO
import time, datetime
now = datetime.datetime.now()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

 #GPIO ports for the 7seg pins
segment8 =  (26,19,13,6,5,11,9,10)

for segment in segment8:
    GPIO.setup(segment, GPIO.OUT)
    GPIO.output(segment, 1)
    #Digit 1
    GPIO.setup(7, GPIO.OUT)
    GPIO.output(7, 0) #Off initially
    #Digit 2
    GPIO.setup(8, GPIO.OUT)
    GPIO.output(8, 0) #Off initially
    #Digit 3
    GPIO.setup(25, GPIO.OUT)
    GPIO.output(25, 0) #Off initially
    #Digit 4
    GPIO.setup(24, GPIO.OUT)
    GPIO.output(24, 0) #Off initially
null = [1,1,1,1,1,1,1]
zero = [0,0,0,0,0,0,1]
one = [1,0,0,1,1,1,1]
two = [0,0,1,0,0,1,0]
three = [0,0,0,0,1,1,0]
four = [1,0,0,1,1,0,0]
five = [0,1,0,0,1,0,0]
six = [0,1,0,0,0,0,0]
seven = [0,0,0,1,1,1,1]
eight = [0,0,0,0,0,0,0]
nine = [0,0,0,0,1,0,0]
def print_segment(charector):
    if charector == 1:
        for i in range(7):
            GPIO.output(segment8[i], one[i])
    if charector == 2:
        for i in range(7):
            GPIO.output(segment8[i], two[i])
    if charector == 3:
        for i in range(7):
            GPIO.output(segment8[i], three[i])
    if charector == 4:
        for i in range(7):
            GPIO.output(segment8[i], four[i])
    if charector == 5:
        for i in range(7):
            GPIO.output(segment8[i], five[i])
    if charector == 6:
        for i in range(7):
            GPIO.output(segment8[i], six[i])
    if charector == 7:
        for i in range(7):
            GPIO.output(segment8[i], seven[i])
    if charector == 8:
        for i in range(7):
            GPIO.output(segment8[i], eight[i])
    if charector == 9:
        for i in range(7):
            GPIO.output(segment8[i], nine[i])
    if charector == 0:
        for i in range(7):
            GPIO.output(segment8[i], zero[i])

    return;
while 1:
    for i in range(61):
        i1=i/10
        i2=i%10
    delay_time = 0.1 #delay to create virtual effect


    GPIO.output(7, 1) #Turn on Digit One
    print_segment (int(i1)) #Print h1 on segment
    time.sleep(delay_time)
    GPIO.output(7, 0) #Turn off Digit One
    GPIO.output(8, 1) #Turn on Digit One
    print_segment (int(i2)) #Print h1 on segment
#Display point Off
    time.sleep(delay_time)
    GPIO.output(8, 0) #Turn off Digit One
```