**HANDOUT 1**

**Installing DHT22 Library to the Raspberry Pi**

From the Terminal: do this

sudo apt-get update

sudo apt-get upgrade

sudo pip3 install --upgrade adafruit-blinka

sudo pip3 install adafruit-circuitpython-dht

|  |  |
| --- | --- |
| #SAVE THIS PROGRAM AS testdht.py  #Library  from time import sleep  import board  import adafruit\_dht  #Components setup  dhtDevice = adafruit\_dht.DHT22(board.D17, use\_pulseio=False)  #Login  while True:  try:  temperature\_c = dhtDevice.temperature  humidity = dhtDevice.humidity  print(temperature\_c,humidity)  sleep(2)  except:  pass | #SAVE THIS PROGRAM AS testdhtfunction.py  #Library  from time import sleep  import board  import adafruit\_dht  #Components setup  dhtDevice = adafruit\_dht.DHT22(board.D17, use\_pulseio=False)  #functions  def readDHT22():  try:  temp\_c = dhtDevice.temperature  hum = dhtDevice.humidity  except:  temp\_c=0  hum=0    return (hum, temp\_c)  #Program  while True:  humidity,temperature = readDHT22()  print(humidity,temperature)  sleep(5) |

**HANDOUT 2**

**LIGHT DEPENDENT RESISTSOR – LDR**

|  |  |
| --- | --- |
| #SAVE THIS PROGRAM as ldrtest.py  #Library  import RPi.GPIO as GPIO  import time  #Component Setup  GPIO.setmode(GPIO.BCM)  resistorPin = 18  #Program  while True:  GPIO.setup(resistorPin, GPIO.OUT)  GPIO.output(resistorPin, GPIO.LOW)  time.sleep(0.1)    GPIO.setup(resistorPin, GPIO.IN)  currentTime = time.time()  diff = 0    while (GPIO.input(resistorPin) == GPIO.LOW):  charging\_time = time.time() - currentTime  diff = charging\_time \* 1000 #convert to milliseconds  print( charging\_time, 'converted to milliseconds = ', diff)  time.sleep(1) | #save this program as ldrfunction.py  #Library  import RPi.GPIO as GPIO  import time  from time import sleep  #Component Setup  GPIO.setmode(GPIO.BCM)  resistorPin = 18  #functions  def getchargingtime():  GPIO.setup(resistorPin, GPIO.OUT)  GPIO.output(resistorPin, GPIO.LOW)  time.sleep(0.1)    GPIO.setup(resistorPin, GPIO.IN)  currentTime = time.time()  diff = 0    while(GPIO.input(resistorPin) == GPIO.LOW):  charging\_time = time.time() - currentTime  diff = charging\_time \* 1000  #charging time convert to milliseconds    return diff #charging time in milliseconds  #Program  while True:  lightintensity=getchargingtime()  print(lightintensity, " in milliseconds")  sleep(1) |

**HANDOUT 3 - TESTING RELAY**

|  |  |
| --- | --- |
| #SAVE THIS PROGRAM AS relaytest.py  #Library  import RPi.GPIO as GPIO  from time import sleep  #settings for components  GPIO.setwarnings(False)  GPIO.setmode(GPIO.BCM)  GPIO.setup(20,GPIO.OUT) #IN1 FAN  GPIO.setup(21,GPIO.OUT) #IN2 LIGHT  #Relay is Active-Low -  #High / True will turn it off  #Low / False will turn it on  #Program Logic  GPIO.output(20,True) #OFF FAN  GPIO.output(21,True) #OFF LIGHT  sleep(5)  while True:  GPIO.output(20,False) #ON FAN  sleep(2)  GPIO.output(20,True) #OFF FAN  sleep(5)  GPIO.output(21,False) #ON LIGHT  sleep(2)  GPIO.output(21,True) #OFF LIGHT  sleep(5) | #TESTING RELAY  #SAVE THIS PROGRAM AS relayfunction.py  #Library  import RPi.GPIO as GPIO  from time import sleep  #settings for components  GPIO.setwarnings(False)  GPIO.setmode(GPIO.BCM)  GPIO.setup(20,GPIO.OUT) #IN1 FAN  GPIO.setup(21,GPIO.OUT) #IN2 LIGHT  #Relay is Active-Low -  #High / True will turn it off  #Low / False will turn it on  #functions  def activateFan():  GPIO.output(20,GPIO.LOW)  def deactivateFan():  GPIO.output(20,GPIO.HIGH)  def activateLight():  GPIO.output(21,GPIO.LOW)  def deactivateLight():  GPIO.output(21,GPIO.HIGH)  #Program Logic  GPIO.output(20,True) #OFF FAN  GPIO.output(21,True) #OFF LIGHT  sleep(5)  while True:  activateFan() #ON FAN  sleep(2)  deactivateFan() #OFF FAN  sleep(5)  activateLight() #ON LIGHT  sleep(2)  deactivateLight() #OFF LIGHT  sleep(5) |

**HANDOUT #4 – CREATING YOUR OWN LIBRARY**

#save program as mylibrary.py

#Library

import RPi.GPIO as GPIO

from time import sleep

import time

from time import sleep

import board

import adafruit\_dht

#components and setting for relay

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(20,GPIO.OUT)

GPIO.setup(21,GPIO.OUT)

#Component Setup for LDR

GPIO.setmode(GPIO.BCM)

resistorPin = 18

#Components setup for DHT22

dhtDevice = adafruit\_dht.DHT22(board.D17, use\_pulseio=False)

#functions

def readDHT22():

try:

temperature\_c = dhtDevice.temperature

temp=temperature\_c

humidity = dhtDevice.humidity

if temp == None:

temp=0

humidity=0

except:

temp=0

humidity=0

return (humidity, temp)

#functions for LDR

def getchargingtime():

GPIO.setup(resistorPin, GPIO.OUT)

GPIO.output(resistorPin, GPIO.LOW)

time.sleep(0.1)

GPIO.setup(resistorPin, GPIO.IN)

currentTime = time.time()

**HANDOUT 4 - CONTINUE**

diff = 0

while(GPIO.input(resistorPin) == GPIO.LOW):

charging\_time = time.time() - currentTime

diff = charging\_time \* 1000 #convert to milliseconds

return diff #charging time in milliseconds

#functions for relay

def activateFan():

GPIO.output(20,GPIO.LOW)

def deactivateFan():

GPIO.output(20,GPIO.HIGH)

def activateLight():

GPIO.output(21,GPIO.LOW)

def deactivateLight():

GPIO.output(21,GPIO.HIGH)

**#HANDOUT 5 – CREATING AN AUTOMATED MONITORING SYSTEM**

**#save this as monitoringsystem.py**

import mylibrary as ml

from time import sleep

#setting the variables

lightthreshold=30

tempthreshold=28

#program logic

ml.deactivateFan()

ml.deactivateLight()

while True:

humidity,temperature=ml.readDHT22()

light=ml.getchargingtime()

print("humidity:",humidity,"Light Intensity:",light,"Temp:",temperature)

if light > lightthreshold:

ml.activateLight()

else:

ml.deactivateLight()

if temperature > tempthreshold:

ml.activateFan()

else:

ml.deactivateFan()

sleep(5)

**#HANDOUT 6 – CREATING AN AUTOMATED MONITORING SYSTEM WITH THINGSPEAK**

1. Import the thingspeak library. On terminal type :

**sudo pip3 install thingspeak**

1. Set up a thingspeak account

**#save this as monitoringthingspeak.py**

import mylibrary as ml

from time import sleep

import time

**import thingspeak**

#setting the variables

lightthreshold=30

tempthreshold=28

**channel\_id=1616752**

**write\_key ='YE95JMMNETI7ZHNG'**

**read\_key='A4JZTVF8P7JQEWVZ'**

#program logic

ml.deactivateFan()

ml.deactivateLight()

while True:

humidity,temperature=ml.readDHT22()

light=ml.getchargingtime()

print("humidity:",humidity,"Light Intensity:",light,"Temp:",temperature)

if light > lightthreshold:

ml.activateLight()

else:

ml.deactivateLight()

if temperature > tempthreshold:

ml.activateFan()

else:

ml.deactivateFan()

**channel = thingspeak.Channel(channel\_id,write\_key)**

**response=channel.update({'field1':light,'field2':temperature,'field3':humidity})**

sleep(5)