EMBEDDED SYSTEMS PROJECT

CB.EN.U4CSE21211- B LOKESH CHANDRA

CB.EN.U4CSE21232- KOLLEPALLI RISHITH PRANAV KUMAR

CB.EN.U4CSE21256- MADDALI SASIDAR

CB.EN.U4CSE21269- Y G RAM DARSHAN REDDY

TITLE :- AUTOMATIC DOOR

CODE:

from machine import Pin, PWM,UART

import utime

from time import sleep

pwm= PWM(Pin(15))

pwm.freq(50)

uart = UART(0,9600)

trigger = Pin(6, Pin.OUT)

echo = Pin(7, Pin.IN)

distance = 0

led=Pin(17,Pin.OUT)

led1=Pin(5,Pin.OUT)

led2=Pin(10,Pin.OUT)

led3=Pin(22,Pin.OUT)

def servo(angle):

c= int((34.45\*angle)+1800)

pwm.duty\_u16(c)

def get\_distance():

global distance

trigger.high()

utime. sleep(0.00001)

trigger.low()

while echo.value() == 0:

start = utime.ticks\_us()

while echo. value() == 1:

stop = utime.ticks\_us()

duration = stop - start

distance = (duration \*0.0343)/2

print(distance,"cm")

return distance

class DHT11:

\_temperature: float

\_humidity: float

def \_init\_(self, pin):

self.\_pin = pin

self.\_last\_measure = utime.ticks\_us()

self.\_temperature = -1

self.\_humidity = -1

def measure(self):

current\_ticks = utime.ticks\_us()

if utime.ticks\_diff(current\_ticks, self.\_last\_measure) < MIN\_INTERVAL\_US and (

self.\_temperature > -1 or self.\_humidity > -1

):

# Less than a second since last read, which is too soon according

# to the datasheet

return

self.\_send\_init\_signal()

pulses = self.\_capture\_pulses()

buffer = self.\_convert\_pulses\_to\_buffer(pulses)

self.\_verify\_checksum(buffer)

self.\_humidity = buffer[0] + buffer[1] / 10

self.\_temperature = buffer[2] + buffer[3] / 10

self.\_last\_measure = utime.ticks\_us()

@property

def humidity(self):

self.measure()

return self.\_humidity

@property

def temperature(self):

self.measure()

return self.\_temperature

def \_send\_init\_signal(self):

self.\_pin.init(Pin.OUT, Pin.PULL\_DOWN)

self.\_pin.value(1)

utime.sleep\_ms(50)

self.\_pin.value(0)

utime.sleep\_ms(18)

@micropython.native

def \_capture\_pulses(self):

pin = self.\_pin

pin.init(Pin.IN, Pin.PULL\_UP)

val = 1

idx = 0

transitions = bytearray(EXPECTED\_PULSES)

unchanged = 0

timestamp = utime.ticks\_us()

while unchanged < MAX\_UNCHANGED:

if val != pin.value():

if idx >= EXPECTED\_PULSES:

raise InvalidPulseCount(

"Got more than {} pulses".format(EXPECTED\_PULSES)

)

now = utime.ticks\_us()

transitions[idx] = now - timestamp

timestamp = now

idx += 1

val = 1 - val

unchanged = 0

else:

unchanged += 1

pin.init(Pin.OUT, Pin.PULL\_DOWN)

if idx != EXPECTED\_PULSES:

raise InvalidPulseCount(

"Expected {} but got {} pulses".format(EXPECTED\_PULSES, idx)

)

return transitions[4:]

def \_convert\_pulses\_to\_buffer(self, pulses):

"""Convert a list of 80 pulses into a 5 byte buffer

The resulting 5 bytes in the buffer will be:

0: Integral relative humidity data

1: Decimal relative humidity data

2: Integral temperature data

3: Decimal temperature data

4: Checksum

"""

# Convert the pulses to 40 bits

binary = 0

for idx in range(0, len(pulses), 2):

binary = binary << 1 | int(pulses[idx] > HIGH\_LEVEL)

# Split into 5 bytes

buffer = array.array("B")

for shift in range(4, -1, -1):

buffer.append(binary >> shift \* 8 & 0xFF)

return buffer

def \_verify\_checksum(self, buffer):

# Calculate checksum

checksum = 0

for buf in buffer[0:4]:

checksum += buf

if checksum & 0xFF != buffer[4]:

raise InvalidChecksum()

pin = Pin(4, Pin.OUT, Pin.PULL\_DOWN)

sensor = DHT11(pin)

t = (sensor.temperature)

h = (sensor.humidity)

print("Temperature: {}".format(sensor.temperature))

print("Humidity: {}".format(sensor.humidity))

while True:

get\_distance()

sleep(2)

if uart.any(): #Checking if data available

data=uart.read() #Getting data

data=str(data)

data=data[2:-5]

print(data)

t = (sensor.temperature)

h = (sensor.humidity)

print("Temperature: {}".format(sensor.temperature))

print("Humidity: {}".format(sensor.humidity))

if 'a' in data:

while True:

get\_distance()

if distance>30:

servo(0)

led.value(0)

led2.value(0)

led3.value(0)

if distance>20 and distance<30:

servo(45)

led.value(0)

led2.value(0)

led3.value(0)

if distance>15 and distance<20:

servo(90)

led.value(0)

led2.value(0)

led3.value(1)

if distance>10 and distance<15:

led.value(0)

led2.value(1)

led3.value(1)

servo(120)

if distance >1 and distance<10:

led.value(1)

led2.value(1)

led3.value(1)

servo(180)

if distance >1 and distance<10 and t> 32:

led.value(0)

led2.value(0)

led3.value(1)

servo(0)

sleep(3)

if uart.any(): #Checking if data available

data=uart.read() #Getting data

data=str(data)

data=data[2:-5]

print(data)

if 'b' in data:

while True:

if uart.any(): #Checking if data available

data=uart.read() #Getting data

data=str(data)

data=data[2:-5]

print(data)

if 'a' in data:

break

if int(data)>150:

led.value(1)

led2.value(1)

led3.value(1)

if int(data)>100 and int(data)<=150:

led.value(0)

led2.value(1)

led3.value(1)

if int(data)>50 and int(data)<=100:

led.value(0)

led2.value(0)

led3.value(1)

if int(data)<=50 :

led.value(0)

led2.value(0)

led3.value(0)

servo(int(data))

CIRCUIT DIAGRAM :-

A circuit board with wires and cables

Description automatically generated

USE CASES :-

An automatic door with a temperature sensor that closes if your temperature is high can be useful in various environments for different reasons. Here are some potential use cases:

1. **Healthcare Facilities:**
   * Hospitals and clinics can use such doors in isolation rooms or areas where contagious patients are treated. The door can automatically close if someone with a high body temperature attempts to enter, helping to prevent the spread of infectious diseases.
2. **Public Transportation:**
   * Airports, train stations, and bus terminals could use these doors at entry points to screen individuals for signs of fever. If someone registers an elevated temperature, the door can remain closed, preventing potential health risks in crowded transportation hubs.
3. **Office Buildings:**
   * In office buildings, such doors could be installed at the entrance or in specific areas to enhance workplace safety. Employees or visitors with a high temperature might be denied access to certain areas until their health status is verified.
4. **Educational Institutions:**
   * Schools and universities can deploy these doors at entrances or in key locations to screen students, faculty, and visitors. If someone exhibits signs of illness, the door can remain closed to protect the health of others on the premises.
5. **Public Spaces:**
   * High-traffic public spaces, such as shopping malls or entertainment venues, could benefit from temperature-sensing doors. This adds an extra layer of safety by identifying and isolating individuals with elevated temperatures.
6. **Manufacturing Facilities:**
   * Industrial environments, especially those with clean room requirements, might use these doors to control access based on health parameters. It helps in maintaining a controlled environment and ensures the well-being of workers.
7. **Residential Spaces:**
   * In smart homes, an automatic door with a temperature sensor could be employed as part of a health monitoring system. If someone within the household displays a high temperature, access to certain areas could be restricted for the safety of others.
8. **Security Checkpoints:**
   * The doors can be used in conjunction with security checkpoints at critical locations. This adds an additional layer of screening to identify individuals with potential health risks before granting access.
9. **Emergency Response Centers:**
   * Critical facilities like emergency response centers or disaster management units could use these doors to safeguard personnel during health emergencies.
10. **Public Events and Venues:**
    * During large gatherings, such as concerts or sports events, these doors can be employed at entry points to screen attendees for elevated temperatures, promoting a safer and healthier environment.