**SAVITRIBAI PHULE PUNE UNIVERSITY**

A Project Report

On

**Child Monitoring System**

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

**BACHELOR OF ENGINEERING**

BY

Pranav Kumar(T150224236)

Saurabh Singh(T150224246)

Sandeep Samant

Prajawal Goswami

Rohan Chougle

Rajat Rawat

Mohit Yadav

Mohit Sharma

Paras Singh Kaphalia

Navjot Singh

**in**

**T.E. COMPUTER ENGINEERING**



**Army Institute of Technology, Pune – 411015**

**2017-18**



ARMY INSTITUTE OF TECHNOLGY, PUNE

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

*This is to certify that the Seminar titled*

Child Monitoring System

has been prepared and presented by

Pranav Kumar (*T150224236*)

*Of Third Year (Computer Engineering)*

*in partial fulfilment of requirement for the award of*

*Degree of Bachelor of engineering in computer Engineering under the University of Pune during the academic year 2017-18*

SEMINAR GUIDE HEAD OF THE DEPARTMENT

Prof. Sushma Shirke Dr. Sunil Dhore

**ACKNOWLEDGEMENT**

It gives me great pleasure in presenting to the mini-project report on “**Child Monitoring system”**.

I am very thankful to Prof. Sushma Shirke for giving me all the help and guidance needed. It is his consistent effort that forced me to complete the work as early as possible. Her love for perfection and interest in detail has supplemented my own quest for knowledge. Her advice, guidance, help, insights, encouragement and regular-reminders were instrumental in shaping my studies.

I extend my sincere gratitude to Prof.S.R. Dhore, Head of Computer Engineering department for creating a competitive environment and for providing us all the essential facilities and encouragement at the department and institute levels.

Pranav Kumar

(T.E. Computer Engg.)

**ABSTRACT**

In India, a survey conducted by the department of health and human services found thatt over 5 million children are victims of child abuse. Not only in India child abuse also occurred all over the world especially in regions of high children population. In this era, monitoring children has become popular and important. The use raspberry pi for monitoring the children can be used to deter crime and prevent ensure comfort of the child. To monitor the children, we create a prototypee using raspberry pi. This method can help parents to monitor the children easily by online with authentication feature. Parents can monitor their children with live video streaming and temperature control. If the any problem is detected, this system will identify it. We also use temperature sensor as an added safety measure and comfort utility. In case any problem is detected, email will send to your profile.

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.NO | . TITLE | Pg.No |
|  | Introduction | 6 |
|  | Problem Statement | 7 |
|  | Hardware Requirement   * 1. Raspberry Pi Specification   2. LDR Specification   3. Sound Sensor Specification   4. Motor Driver Specification   5. LM35 Temperature Sensor | 8  8-10  11-12  12-14  14-16  16-17 |
|  | Software Requirement |  |
|  | UML Diagramme   * 1. Activity-Timing Diagramme   2. Use-Case Diagramme   3. IOT-4 Deployment Template   4. Functional Group Specification   5. Operational View Specification | 18  18-19  19  20  21  21 |
|  | Algorithmns |  |
|  | Steps | 23 |
|  | Programme Codes   * 1. Client-Server   2. Sound   3. LDR   4. Temprature   5. Vedio-Client   6. Vedio-Server | 24-25  26  26-27  28-30  30-32  33-36 |
|  | Conclusion | 37 |
|  | Team Members | 38 |
|  |  |  |

1. **INTRODUCTION**

In India, with the rapid development of urbanization and industrialization, rural resident population gradually reduces. More and more children have started to live in cities. On one hand cities can provide a more favorable environment for children’s living and learning, on the other hand they bring a huge challenge to the growth and safety of children because of the complex work environment of cities, which has resulted in both parents working and no one present for the child. A 4-5-year-old child always needs someone to keep an eye out for him/her since children are inherently curious, active and do not often posses the knowledge of safety measures and requirements.

In India especially, there are barely any systems which focus on child monitoring and their safety. The main purpose household monitoring systems, if any, is to guard against thieves or criminals. There is no system dedicated for young children who are in constant need of a parent’s attention.In India baby-sitters and nannies are usually employed. Thus, a monitoring system with a video feed becomes even more important to prevent any misunderstandings between parents and baby-sitters.Thus, in such a complex urban environment it has become necessary for a monitoring system for the children of the working families so that the future of the country is given proper attention they require.

This will not only help the children, it will also help the parents to have a sense of comfort and security for their child.

Monitoring technology to provide comfort and a sense of security to parents. Innovations and developments in the field of computers have arouse rapidly over time and getting more sophisticated, from small to large. One of the results of computer development is Raspberry Pi. Raspberry Pi is a computer with a small size, the size no larger than a credit card, and with low power, 5V.Based on the data obtained, a prototype of child monitoring system deal with the real-time video streaming and motion detection is required. We are using the Raspberry Pi microcomputer whichinstalled with video, temperature control and sound sensors, a prototype of child monitoring system can fulfill the parent needs to monitor their children. To have parents easy access to monitor their children and prevent misapplication by unauthorized people, we can view the output via online through website and authentication features

**2.PROBLEM STATEMENT**

The implementation of a child monitoring system based on Raspberry Pi and various sensors to help guardians acquire whether their children are safe and comfortable or not. This system judges the safety level and the comfort level using camera, sound sensors and temperature sensors.

1. **HARDWARE REQUIREMEN**

**3.1 RASPBERRY PI :**

A Raspberry Pi is a credit card-sized computer originally designed for education, inspired by the 1981 BBC Micro. Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at the pre-university level .

**3.1.1 Specifications**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Model A** | |  |  | **Model B** |  |  | **Model B+** | |  |  |
|  |  | |  |  |  |  |  |  |  |  |  |
| **Target price:** | US$25 | |  |  | US$35 |  |  |  |  |  |  |
|  |  | | |  | | | |  | | |  |
| **SoC:** | Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and single | | | | | | | | | | USB |
|  | port) |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | | |  |  | | |  | |
| **CPU:** | 700 | MHz | ARM1176JZF-S | | | core | (ARM11 family, | | | ARMv6 | |
|  | instruction | | set) | |  |  |  |  |  |  |  |
|  |  | |  | | |  |  | |  |  |  |
| **GPU:** | Broadcom | | VideoCore | | | IV | @ | | 250 |  | MHz |
|  |  |  |  | |  | | | | | | |
| **Memory** | 256 | MB | (shared |  | 512 MB (shared with GPU) as of 15 October | | | | | | |
| **(SDRAM):** | with GPU) | |  |  | 2012 |  |  |  |  |  |  |
|  |  |  |  |  |  | | |  | | | |
| **USB 2.0 ports:** | 1 | (direct | from |  | 2 (via the on-board 3- | | | 4 (via the on-board 5- | | | |
|  | BCM2835 chip) | | |  | port USB hub) | |  | port USB hub) | | |  |
|  |  | | |  |  |  | |  |  |  | |
| **Video input:** | 15-pin MIPI camera | | | | interface | (CSI) connector, | | | used | with the | |
|  | Raspberry Pi Camera Addon. | | | | |  |  |  |  |  |  |
|  |  | | | | | | | | | | |
| **Video outputs:** | Composite RCA (PAL and NTSC) –in model B+ via 4-pole 3.5 mm | | | | | | | | | | |
|  | jack, HDMI (rev 1.3 & 1.4), raw LCD Panels via DS | | | | | | | |  |  |  |
|  |  | | | | | | | | | |  |
| **Audio outputs:** | 3.5 mm jack, HDMI, and, as of revision 2 boards, I²S audio | | | | | | | | | | (also |
|  | potentially | | for audio input) | | |  |  |  |  |  |  |
|  |  | | | | | | |  | |  |  |
| **Onboard** | SD / MMC / SDIO card slot (3.3 V card | | | | | | | MicroSD | |  |  |
| **storage:** | power support only) | | | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

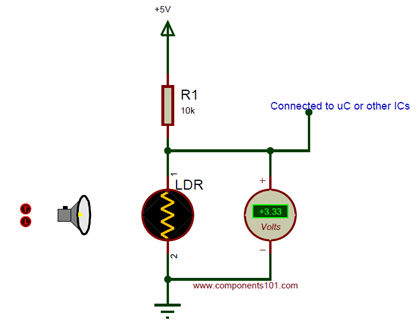
14

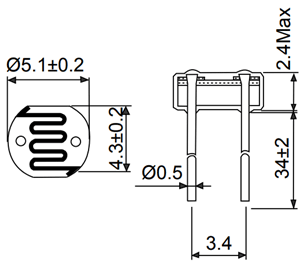
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Onboard** | None | 10/100 Mbit/s Ethernet | | (8P8C) USB adapter |
| **network:** |  | on the third/fifth port of the USB hub | | |
|  |  |  |  |  |
| **Low-level** | 8× GPIO, UART, I²C bus, SPI bus with two | |  | 17× GPIO |
| **peripherals:** | chip selects, I²S audio +3.3 V, +5 V, ground | |  |  |
|  |  |  |  |  |
| **Power ratings:** | 300 mA (1.5 W) | 700 mA (3.5 W) |  | 600 mA (3.0 W) |
|  |  |  |  |  |
| **Power source:** | 5 V via MicroUSB or GPIO header | | |  |
|  |  | | | |
| **Size:** | 85.60 mm × 56 mm (3.370 in × 2.205 in) – not including protruding | | | |
|  | connectors |  |  |  |
|  |  |  |  |  |
| **Weight:** | 45 g (1.6 oz) |  |  |  |
|  |  |  |  |  |

*Table 1 Specifications*

**3.1.2 LDR Specification :**

The Light Dependent Resistor (LDR) is just another special type of Resistor and hence has no polarity. Meaning they can be connected in any direction. They are breadboard friendly and can be easily used on a perf board also. The symbol for LDR is just as similar to Resistor but adds to inward arrows as shown above. The arrows indicate the light signals.

****

****

LDR Features

* Can be used to sense Light
* Easy to use on Breadboard or Perf Board
* Easy to use with Microcontrollers or even with normal Digital/Analog IC
* Small, cheap and easily available
* Available in PG5 ,PG5-MP, PG12, PG12-MP, PG20 and PG20-MP series

**3.1.3 Sound Sensor Specifications:**

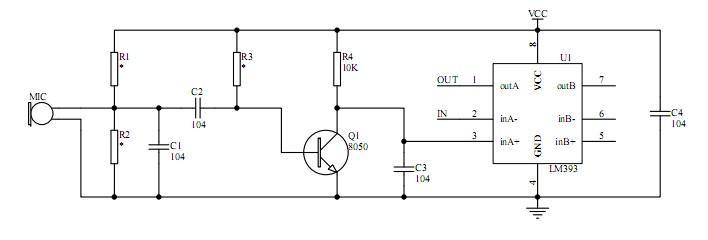
The sound sensor module provides an easy way to detect sound and is generally used for detecting sound intensity. This module can be used for security, switch, and monitoring applications. Its accuracy can be easily adjusted for the convenience of usage.

It uses a microphone which supplies the input to an amplifier, peak detector and buffer. When the sensor detects a sound, it processes an output signal voltage which is sent to a microcontroller then performs necessary processing.

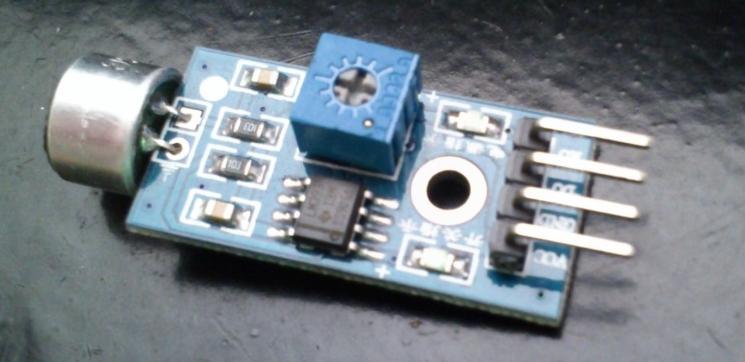
**Specifications**

* Operating voltage 3.3V-5V
* Output model: digital switch outputs (0 and 1, high or low level)
* With a mounting screw hole

**Schematic Diagram**

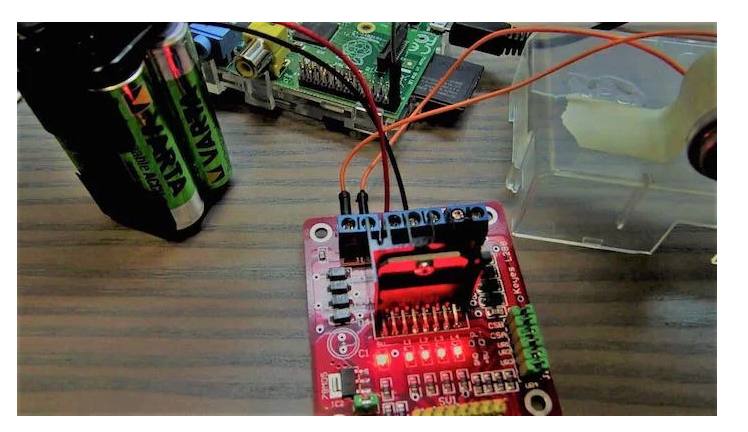
****

**Pin Configuration**

****

1. VCC: 3.3V-5V DC
2. GND: ground
3. DO: digital output

**3.1.4 L298 motor driver specification :**



L298 is known as a dual bidirectional motor driver, which is based on dual H-Bridge Motor driver IC. This circuit allows you to control two DC motors independently in either direction.

It is a commonly used component for prototypes and hobbyist projects, as it is easy to use and interface the L298 with a Raspberry Pi or an Arduino. Other than its minimal design, it also provides an onboard 5V regulator that you can use to power your 5V circuits very conveniently.

There are many L298 based motor driver modules out there in the market and you can use any of the locally available L298 based motor drivers because they all are essentially the same. I used an L298 breakout board, which makes setup a little easier. You can find several different L298 breakout boards online.

For communication, we will use a simple serial communication over USB cable. 

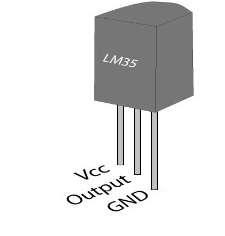
So, let’s get started.

* Connect IN1 on the L298 to the Raspberry Pi’s pin number 26.
* Connect IN2 on the L298 to Raspberry Pi’s pin number 20.
* Connect the ENA and 12-volt pin to a 9-volt battery.
* Make sure the grounds of the battery, Raspberry Pi, and L298 are common.

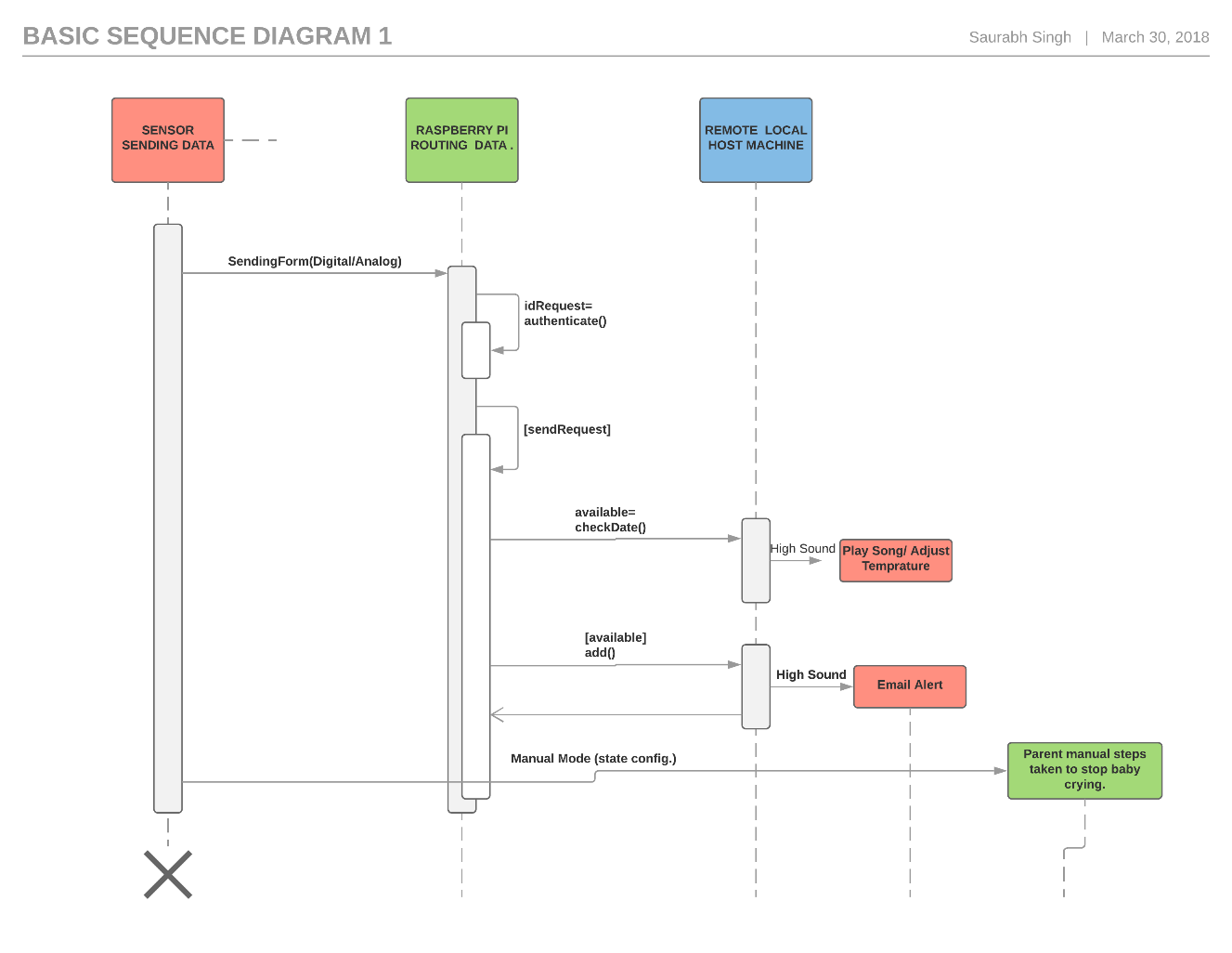
**3.1.5 IC LM35 Temperature Sensor :**

The LM35 is a popular and inexpensive temperature sensor. It provides an output voltage of 10.0mV for each degree Centigrade of temperature from a reference voltage. The output of this device can be fed to A/D Converter; any microcontroller can be interfaced with any A/D Converter for reading and displaying the output of LM35. The circuit should be designed, so that output should be at 0V when the temperature is 0 degrees Centigrade and would rise to 1000mV or 1.0V at 100 degrees Centigrade. To get the temperature value accurately, output voltage must be multiplied with 100. For example if you read 0.50V that would be 50 degrees Centigrade.

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power 2 Applications supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package.



1. **SOFTWARE REQUIREMENT**
2. Python IDLE ( 2.7 or above)
3. Anaconda (2.7 or above)
4. Web App (Html, CSS, JS)
5. NodeJs
6. Python Scripts
7. MongoDB
8. **UML DIAGRAMS**



Activity-Timing Diagram

**A close up of a logo

Description generated with high confidence**

**9 DESIGN METHODOLOGY**

**Purpose** : Develop a smart classroom system which allows us to control the attendance of the students using the web application, and provides smart teaching and power saving methodology.

**Behaviour** : Our system is capable of operating in auto and manua mode.

**System Management Requirement** : System provides remote monitoring and control functions.

**Data Analysis Requirement** : System performs local and cloud analysis of data.

**Security Requirements** : Authentication for admin dashboard.

* 1. **Process Model Specification :**

Attendence Access

Auto

mode

Admin

Teacher

Statistics Details

Register form

Attendence Mark

Manual

Total Attendence

Start Of Process

Decision Box

State

* 1. **DOMAIN MODEL SPECIFICATION**

**Physical Entity** : Discrete and identifiable entities like classroom, students, teachers, administration, classroom power supply, projectors.

**Virtual Entities** : Virtual entities exist for students, classroom power supply, slides used for teaching.

**Device** : Raspberry Pi, camera, ultrasonic sensor.

**Resources** :

On Device resources : Raspbian OS, node server.

Network Resources : Google cloud, Firebase database.

**Services** :

Service to mark attendance(Auto/Manual)

Service to register students(Auto/Manual)

Service to switch off power supply(Auto/Manual)

Service to perform slide transitions(Auto/Manual)

* 1. **Information Model Specification**

Virtual Entity : Presentation

Attribute : State

Virtual Entity : Classroom Power

Absent

Present

Attribute : Attendance

Virtual Entity : Students

Attribute : Slide transition

On

On

On

On

* 1. **SERVICE SPECIFICATIONS**

Power Management

Auto

mode

Automatic Mode

Traditional Mode

Attendance

Slide Transitions

Attendence

Manual

Power Management

Start Of Process

Decision Box

State

Slide Transitions

Virtual Entity : Presentation

Attribute : State

Virtual Entity : Classroom Power

Absent

Present

Attribute : Attendance

Virtual Entity : Students

Attribute : Slide transition

On

On

On

On

**A screenshot of a cell phone

Description generated with high confidence**

**A screenshot of a cell phone

Description generated with high confidence**

**A screenshot of a cell phone

Description generated with very high confidence**

* 1. **DEVICE AND COMPONENT INTEGRATION**

1. Raspberry Pi 3
2. Ultrasonic sensor HC-SR04
3. Qualcomm USB Camera
   1. **APPLICATION DEVELOPMENT**

Firebase DB

Web App

Central Server

Pi Server

Pi Server

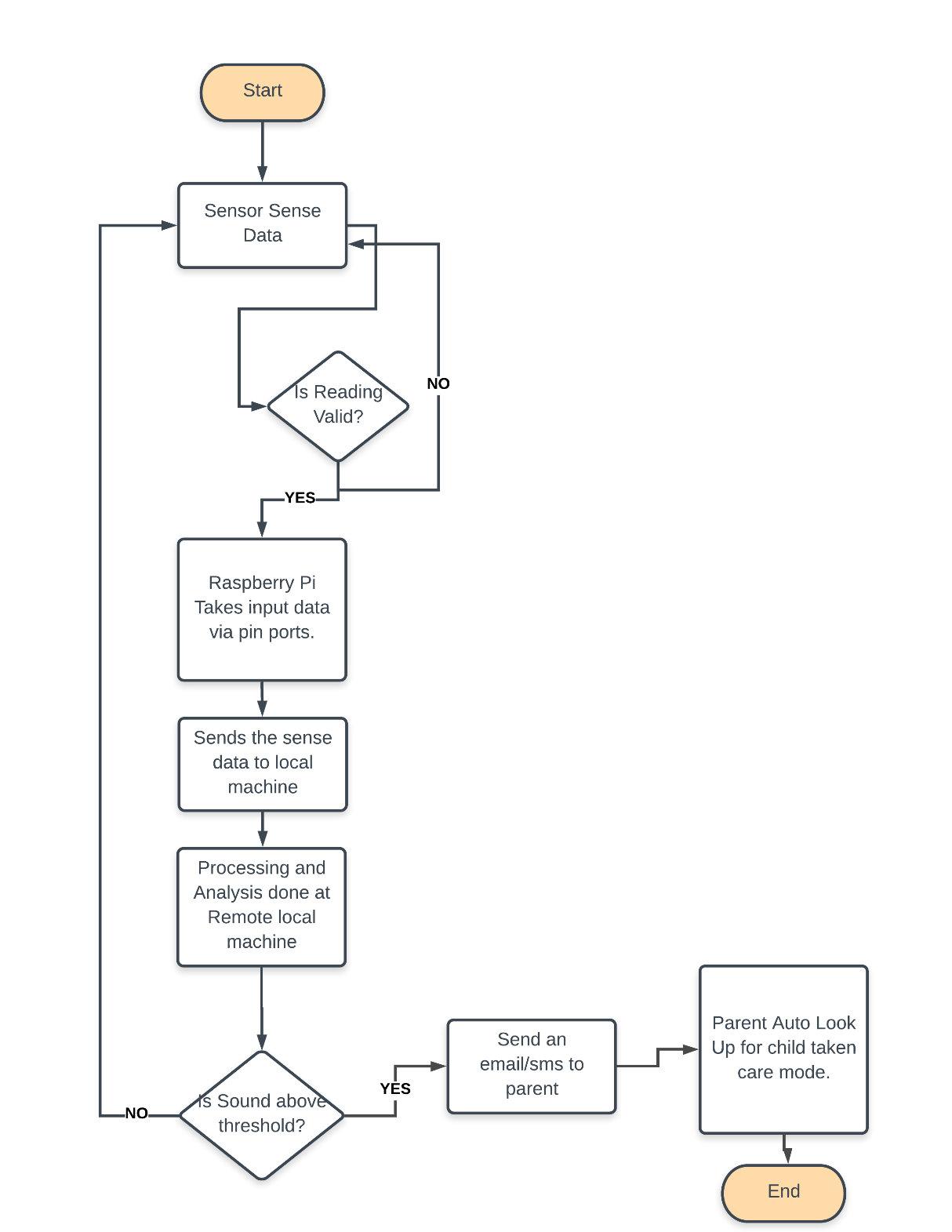
Pi Server

Pi Server

Pi Server

Cloud Server

1. **ALGORITHM**
2. **STEPS**



1. **PROGRAM**:

**Client:**

import cv2

import socket

import pickle

import struct

cap = cv2.VideoCapture(0)

clientsocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

clientsocket.connect(('localhost', 8765))

while True:

ret,frame = cap.read()

data = pickle.dumps(frame)

clientsocket.sendall(struct.pack("L", len(data)) + data)

**Server:**

import socket

import cv2

import pickle

import struct

HOST = '0.0.0.0'

PORT = 8765

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

print 'Socket created'

s.bind((HOST, PORT))

print 'Socket bind complete'

s.listen(1)

print 'Socket now listening'

conn, addr = s.accept()

data = ""

payload\_size = struct.calcsize("L")

while True:

while len(data) < payload\_size:

data += conn.recv(131072)

packed\_msg\_size = data[:payload\_size]

data = data[payload\_size:]

msg\_size = struct.unpack("L", packed\_msg\_size)[0]

while len(data) < msg\_size:

data += conn.recv(131072)

frame\_data = data[:msg\_size]

data = data[msg\_size:]

frame=pickle.loads(frame\_data)

#print frame.size

cv2.imshow('frame', frame)

cv2.waitKey(1)

**Sound:**

import pyaudio

import time

import numpy as np

from matplotlib import pyplot as plt

import scipy.signal as signal

CHANNELS = 1

RATE = 44100

p = pyaudio.PyAudio()

fulldata = np.array([])

dry\_data = np.array([])

def main():

stream = p.open(format=pyaudio.paFloat32,

channels=CHANNELS,

rate=RATE,

output=False,

input=True,

stream\_callback=callback)

stream.start\_stream()

while stream.is\_active():

time.sleep(10)

stream.start\_stream()

numpydata = np.hstack(fulldata)

print(max(numpydata))

stream.close()

**LDR:**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

#define the pin that goes to the circuit

pin\_to\_circuit = 7

def rc\_time (pin\_to\_circuit):

count = 0

#Output on the pin for

GPIO.setup(pin\_to\_circuit, GPIO.OUT)

GPIO.output(pin\_to\_circuit, GPIO.LOW)

time.sleep(0.2) #discharging time of the capacitor

#Change the pin back to input

GPIO.setup(pin\_to\_circuit, GPIO.IN)

#Count until the pin goes high

while (GPIO.input(pin\_to\_circuit) == GPIO.LOW):

count += 1

return count

#Catch when script is interrupted, cleanup correctly

try:

# Main loop

while True:

time.sleep(30)

print rc\_time(pin\_to\_circuit)

except KeyboardInterrupt:

pass

finally:

GPIO.cleanup()

**Temperature:**

#import modules

import time

import RPi.GPIO as GPIO

#pin definitions

ledR = 2

ledG = 3

ledB = 5

#GPIO setup

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(ledR, GPIO.OUT)

GPIO.setup(ledG, GPIO.OUT)

GPIO.setup(ledB, GPIO.OUT)

#set initial LED states

GPIO.output(ledR, 1)

GPIO.output(ledG, 1)

GPIO.output(ledB, 1)

#main loop

try:

while 1:

tempStore = open("/sys/bus/w1/devices/28-0315902106ff/w1\_slave") #change this number to the Device ID of your sensor

data = tempStore.read()

tempStore.close()

tempData = data.split("\n")[1].split(" ")[9]

temperature = float(tempData[2:])

temperature = temperature/1000

print temperature

if temperature < 20: #change this value to adjust the 'too cold' threshold

GPIO.output(ledR, 1)

GPIO.output(ledG, 1)

GPIO.output(ledB, 0)

if temperature > 20 and temperature < 24: #change these values to adjust the 'comfortable' range

GPIO.output(ledR, 1)

GPIO.output(ledG, 0)

GPIO.output(ledB, 1)

if temperature > 24: #change this value to adjust the 'too hot' threshold

GPIO.output(ledR, 0)

GPIO.output(ledG, 1)

GPIO.output(ledB, 1)

time.sleep(1)

except KeyboardInterrupt:

GPIO.cleanup()

print ("Program Exited Cleanly")

**Video-Client:**

import sys

import socket

import \_thread

import time

import signal

import cv2

# Create socket and listen on port 5005

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

server\_socket.bind(("", 5005))

server\_socket.listen(5)

cam = cv2.VideoCapture(1)

def signal\_handler(signal=None, frame=None):

exit(0)

def capture(client\_socket):

while 1:

try:

ret, frame = cam.read()

data = cv2.imencode('.jpg', frame)[1].tostring()

send(client\_socket,data)

except KeyboardInterrupt:

signal\_handler()

def send(c, data):

try:

c.send(data)

c.send(b"END!") # send param to end loop in client

except socket.error:

server\_socket.close()

exit(0)

while 1:

try:

client\_socket, address = server\_socket.accept()

print

"Conencted to - ", address, "\n"

capture(client\_socket)

except socket.timeout:

continue

except KeyboardInterrupt:

server\_socket.close()

exit(0)

**Video-Server:**

#!/usr/bin/python

import socket

import cv2

import numpy

import random

import sys

host = "localhost" # e.g. localhost, 192.168.1.123

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

client\_socket.connect((host, 5005))

name = str(random.random()) # gives random name to create window

def rcv():

data = b''

while 1:

try:

r = client\_socket.recv(90456)

if len(r) == 0:

exit(0)

a = r.find(b'END!')

if a != -1:

data += r[:a]

break

data += r

except Exception as e:

print(e)

continue

nparr = numpy.fromstring(data, numpy.uint8)

if nparr.size!=0:

frame = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

else:

return None

if type(frame) is type(None):

pass

else:

return frame

# try:

# cv2.imshow(name,frame)

# if cv2.waitKey(10) == ord('q'):

# client\_socket.close()

# sys.exit()

# except:

# client\_socket.close()

# exit(0)

# while 1:

# rcv()

**Video:**

import cv2

import numpy as np

import time

from s import rcv

from darkflow.net.build import TFNet

options = {

'model': 'cfg/tiny-yolo-voc.cfg',

'load': 'bin/tiny-yolo-voc.weights',

'threshold': 0.2,

'gpu': 0.7

}

tfnet = TFNet(options)

colors = [tuple(255 \* np.random.rand(3)) for \_ in range(10)]

while True:

stime = time.time()

# ret, frame = capture.read()

frame = rcv()

if type(frame) is type(None):

continue

results = tfnet.return\_predict(frame)

if 1:

for color, result in zip(colors, results):

tl = (result['topleft']['x'], result['topleft']['y'])

br = (result['bottomright']['x'], result['bottomright']['y'])

label = result['label']

confidence = result['confidence']

if label == "bottle":

label = "Baby"

text = '{}: {:.0f}%'.format(label, confidence \* 100)

frame = cv2.rectangle(frame, tl, br, color, 5)

frame = cv2.putText(

frame, text, tl, cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 0, 0), 2)

cv2.imshow('frame', frame)

print('FPS {:.1f}'.format(1 / (time.time() - stime)))

if cv2.waitKey(1) & 0xFF == ord('q'):

break

capture.release()

cv2.destroyAllWindows()

1. **CONCLUSION**

In this project we were successfully able to develop a child monitoring system that can monitor the condition of a child. This monitored information can be securely accessed by the parent remotely on a web interface.

In future this system can include a fire-alarm , fire sprinkerls to make the life of the child more secure. Further more this sytem can be added to home automation system to further improve the life quality.

**Group Members**

1. **Sandeep Samant**
2. **Prajwal Goswami**
3. **Rohan Chougule**
4. **Rajat Rawat**
5. **Saurabh Singh**
6. **Mohit Yadav**
7. **Mohit Sharma**
8. **Navjot Singh**
9. **Paras Singh Kaphalia**
10. **Pranav Kumar**