**A picture containing tableware, clipart, plate, dishware

Description generated with very high confidence**

**A Project Report**

**on**

**IoT based Cradle system using SIDS monitor**

**For the award of**

**ONTARIO GRADUATE CERTIFICATE**

Submitted by

**Rohan Yadav - C0773871**

**Shahrukh Padaniya - C0769542**

**Swapnil Sevak - C0777195**

**Vandana Eaga - C0777215**

Under the guidance of

**Dr. Mike Aleshams**

**ESE-4009 Embedded System Design Project**

(May 2021 - Aug 2021)

**Abstract**

The aim of this project was to develop an IoT based system to control and monitor cradles for babies. The user would be able to control parameters such as music player, swing control as well as monitor things like motion detection, pulse oximeter, camera, humidity and temperature.

We have used Raspberry Pi and some sensors in the project and the system runs on Raspbian OS. The project is integrated with the webpage, so that the user can use this project remotely through a webpage.

The project will be ready to use in plug and play mode. We have tested each component and their result before integrating it with the webpage. Finally, to support early users, we have created a user guide, which will help them to understand and use our system.

**Acknowledgments**

We would like to express our deepest appreciation to everyone at Embedded Systems Design Engineering Class in Lambton College in Toronto who provided us the possibility to complete this project. A special gratitude to project guide, Dr. Mike Aleshams whose engagement and contribution helped us coordinate our project and filing the final project report. His teaching throughout the course helped us to develop this project.

We would also like to extend our appreciation to Dr. Takis Zourntos, Course Coordinator, Embedded Systems Design Engineering for providing necessary cooperation during troubleshooting times.

Lambton College in Toronto is a great place to study and work, largely because of supporting faculty members. Thanks to all my colleagues in college for supporting and encouraging our idea.

ROHAN YADAV - C0773871

SHAHRUKH PADANIYA - C0769542

SWAPNIL SEVAK - C0777195

VANDANA EAGA - C0777215

Table of Contents

**List of Figures -----------------------------------------------------------------------------------------6**

**Chapter I - Introduction ----------------------------------------------------------------------------7**

**Overview 7**

Problem Statement--------------------------------------------------------------------------------------------7

Goal and Objective--------------------------------------------------------------------------------------------8

Scope of Project-----------------------------------------------------------------------------------------------8

Outcomes and Benefits---------------------------------------------------------------------------------------9

Facilities and Resources--------------------------------------------------------------------------------------9

Procedure and Methodology---------------------------------------------------------------------------------9

**Chapter II – Literature Review--------------------------------------------------------------------10**

**Chapter III – Requirement and Analysis--------------------------------------------------------12**

Hardware------------------------------------------------------------------------------------------------------12

Software--------------------------------------------------------------------------------------------------------12

Power Requirement-------------------------------------------------------------------------------------------13

**Chapter IV – Design---------------------------------------------------------------------------------14**

Raspberry Pi interfacing-------------------------------------------------------------------------------------14

Pulse Oximeter interfacing----------------------------------------------------------------------------------14

Humidity Sensor interfacing--------------------------------------------------------------------------------15

PIR sensor interfacing----------------------------------------------------------------------------------------17

GSM module interfacing with Raspberry Pi--------------------------------------------------------------18

Servo Motor and Motor driver------------------------------------------------------------------------------19

USB speaker interfacing-------------------------------------------------------------------------------------19

**Chapter V – Implementation and Test-----------------------------------------------------------21**

Raspberry pi and PIR sensor-------------------------------------------------------------------------21

Raspberry pi and Pulse - oximeter MAX30102---------------------------------------------------21

Raspberry pi and USB camera-----------------------------------------------------------------------23

Raspberry pi and GSM Module----------------------------------------------------------------------24

Raspberry pi and servo motor------------------------------------------------------------------------25

Raspberry pi and Humidity and Temperature sensor with DHT11------------------------------25

Raspberry pi and USB Speaker----------------------------------------------------------------------27

Raspberry pi and Web server-------------------------------------------------------------------------28

Real time operation------------------------------------------------------------------------------------29

Final Prototype-----------------------------------------------------------------------------------------44

**Chapter VI – Evaluation----------------------------------------------------------------------------47**

Introduction---------------------------------------------------------------------------------------------------47

Minimum Requirements-------------------------------------------------------------------------------------47

Troubleshooting----------------------------------------------------------------------------------------------47

**Chapter VII – Conclusion--------------------------------------------------------------------------48**

Future Work---------------------------------------------------------------------------------------------------48

**Chapter VIII – User’s Guide-----------------------------------------------------------------------49**

**References----------------------------------------------------------------------------------------------53**

**List of Figures**

|  |  |
| --- | --- |
| Figure No. | Description |
| 2.1 | Project Timeline |
| 3.1 | Block Diagram of IoT based Cradle using SIDS monitor |
| 4.1 | Raspberry Pi interfacing |
| 4.2 | Pulse oximeter |
| 4.3 | Pinout of pulse oximeter |
| 4.4 | DHT11 humidity sensor |
| 4.5 | Pinout of DHT11 |
| 4.6 | PIR sensor |
| 4.7 | Pinout of PIR sensor |
| 4.8 | GSM module |
| 4.9 | Motor driver IC |
| 4.10 | USB Camera |
| 5.1 | PIR sensor interfacing with Raspberry pi |
| 5.2 | I2C interfacing enabled |
| 5.3 | PIR sensor output on Raspberry pi |
| 5.4 | Video Capture by Guvcview |
| 5.5 | SIM868 interfacing Raspberry pi |
| 5.6 | Servo motor and L298N interfacing Raspberry pi |
| 5.7 | DHT11 interfacing Raspberry pi |
| 5.8 | Output of DHT11 interfacing Raspberry pi |
| 5.9 | Output of USB speaker interfacing Raspberry pi |
| 5.10 | Install nginx |
| 5.11 | RPi Web page |

**Chapter I**

**Introduction**

**Overview**



IOT based cradle using SIDS monitor system works as a helping assistant of a mother who specializes in baby’s health monitoring. This system provides alert in an emergency case, if the baby stops breathing or heart rate is not detected, so it will grab the attention of parents so that they can take action immediately. In addition to that, there will be a video camera attached to it, so that parents can monitor remotely. Cry detection circuits will analyse cry patterns and give information to parents if crying activity persists for a long time and activates swing of the cradle. Wetness will be detected if the baby urinates in the bed and provides an alert to parents. All sensors are always connected to the internet and GSM Module provides information through user friendly android apps. This app will have a feature of swing control when the baby is crying, switching to video surveillance.

This product costs around 400-500 dollars. The users can easily access through the webpage and know the activities of the baby in real time and get the emergency alerts even though there is no internet.

**Problem Statement**



In today's advanced technology there are some products that measure an infant's oxygen and heart rate of a baby, there are other products that just function as a surveillance system. But there are few products that combine all the features. It is time to develop a project which will be very useful to the parents. Existing technology is limited to one or two features they lack many features like surveillance system, playing melodious sounds when baby cries, automatic swing for cradle and notification to guardians through SMS.

**Goal and Objectives**

* **Goal**

Develop a IoT based Cradle system using SIDS monitor with a plug and play mode and it should be user friendly by the end of summer 2021 academic term.

* **Objective**
* Interfacing the Raspberry pi with USB camera, speaker,DHT11 sensor,PIR sensor,Pulse oximeter,servo motor and GSM module.
* Creating a webpage and operating in real time.
* Creating a prototype to design the cradle.

**The Scope of the Project**

* **Deliverable**
* A smart baby cradle with a USB webcam and Speaker
* a bandaid placed to the foot of baby to detect heart rate
* Power adapter
* Sensors for detecting Humidity and temperature.
* Security system that provides alerts if someone is around the baby.
* **Milestones**
* June 10, 2021 - Testing components
* June 14,2021 - Setting up Raspberry pi
* June 21,2021 - Video Recorder interfacing
* June 24,2021 - Pulse oximeter interfacing
* June 25,2021 - Humidity sensor interfacing
* June 30,2021- PIR sensor interfacing
* July 5,2021 - GSM module interfacing
* July 8,2021 - Motor interfacing
* July 12,2021 - Speaker interfacing
* July 19,2021 - Cloud storage management
* July 22,2021 - Schematic capture design
* July 26,2021 - PCB layout design
* July 29,2021 - Web designing
* August 5, 2021 - Code integration
* August 9,2021 - Real - time operation
* August 12,2021 - Power management

**Limitations**

* Users can access website locally because we don't own any domain to publish website.
* The baby has to wear the bandaid when placed in the cradle to monitor the heart rate.
* Pulse oximeter generate false alarms sometimes due to light interference.

**Outcomes and Benefits**

* Improve childcare
* Provide helping hand to working women
* Alert Parents in emergency case of SIDS
* Provide live footage of the baby using a USB camera.
* Entertain through music and swing control.
* Alert parents when the baby's bed is wet.
* Give all the attention to the baby that he/she needs.

**Facilities and Resources**

* **Internet**
  + Various resources from the internet help to develop final software for the project.
* **Intellectual Resources**
  + DEREK MOLLY. (2016). Exploring Raspberry pi. WILEY.
* **Cellular facility:**
* Used fido sim that supports GSM frequency for alerting parents in emergency cases.

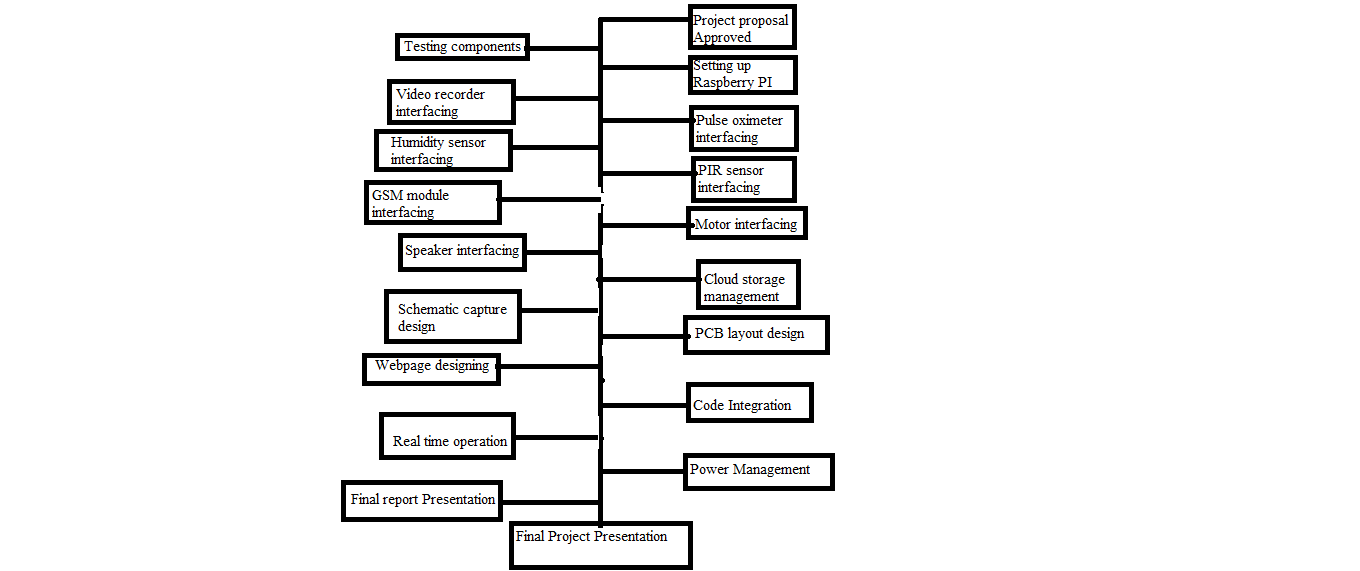
**Procedure and Methodology**

1. Install a suitable Raspian image for Raspberry pi.
2. Attach a projector and Install VNC viewer or PuTTy to access raspberry pi.
3. Attach Pir sensor,servo motor through motor driver to GPIO
4. For pulse oximeter, Attach arduino Uno and enable serial communication and get data of pulse oximeter which is attached to arduino.
5. Attach USB camera and USB speaker to raspberry pi.
6. Attach SIM868 GSM module to USB port.

**Chapter II**

**Literature Review**

We started research about the project since, previous semester. So from this project our main motive is to solve the problem of working parents. According to many published articles there are a huge number of deaths of infant’s just because of lack of attention towards them.



*Figure 2.1 Project Timeline*

So, we decided to do something in that direction and planned to make an IOT based cradle system which can monitor all the activities of babies and can frequently check their oxygen level and heart bits to avoid any casualties.

Then we start doing research about any related projects to get a rough idea about the process and perfect execution of the project. But there is no project or prototype related to our idea.

It took more than 3 months to develop such a prototype which is in working condition.

There is no reference product to give an exact idea about our project. But this is a lot more advanced than conventional cradles. This product will be used even for wetness detection and also to check the presence of a baby in a cradle.

**Chapter III**

**Requirements / Analysis**

**Hardware** 

This portable projector requires a touchscreen display, host processor which can run Debian Linux and Wi-Fi, Bluetooth capabilities. To remove PCB design and wiring, the breadboard team decided to use capes and make the design modular.

Product fits for both entertainment and professional purposes, you can watch a movie at night or can give your presentation on the go, and the projector is expected to work for at least 2.5 hours of movie time or 3.5 hours if the screen is turned off. The prototype is equipped with a resistive touchscreen which can be replaced by capacitive on product development.

Following is a list of required hardware for this project:

* Microprocessor - RASPBERRY PI 3 MOD B+ BCM2837B0
* PIR sensor - HC-SR501
* USB camera and microphone - Logitech C920
* Speaker - USB mini speaker from LIELONGREN
* Humidity sensor - DHT22
* Electric motor - Servo Motor
* Motor Driver - L298N
* Pulse Oximeter - MAX30102
* GSM/GPRS Module - SIM868

**Software** 

Considering software requirements, for developing a prototype team will need a Linux machine or a Windows PC with Putty installed. We decided to forward with a Debian image for Beaglebone black from Beagleboard.org. An Android image was planned to install once tests on Debian images are completed successfully.

Following is a list of required software for this project:

* Raspberry Pi OS (Raspbian)
* C language, C++
* Kicad
* NGINX Web Server

**Power Requirement** 

There are three main power consuming components. Their power requirements are as follows:

1. Beaglebone Black / Wireless – 50mA @ 5V.
2. LCD touchscreen cape – 2000mA @ 5V.
3. DLP 2000 Projector Cape – 320mA @5V for 20 Lumens.

So estimated power requirements for full product assembly will be 2500mA @ 5V for 1 hour of operation.

**Block Diagram**

Raspberry pi

Pulse Oximeter

Humidity sensor

Motor driver

Speaker

Electrical motor

Speaker driver

Video recorder

Power management

Cloud

PIR sensor

WiFi Module

Web

Server

GSM

*Figure 3.1 Block Diagram of IoT based Cradle using SIDS monitor*

**Chapter IV**

**Design**

**Raspberry Pi interfacing**

Raspberry Pi interfacing is the initial stage after testing all the components. It is a very important part as well because many components will be connected, controlled and operated through it.

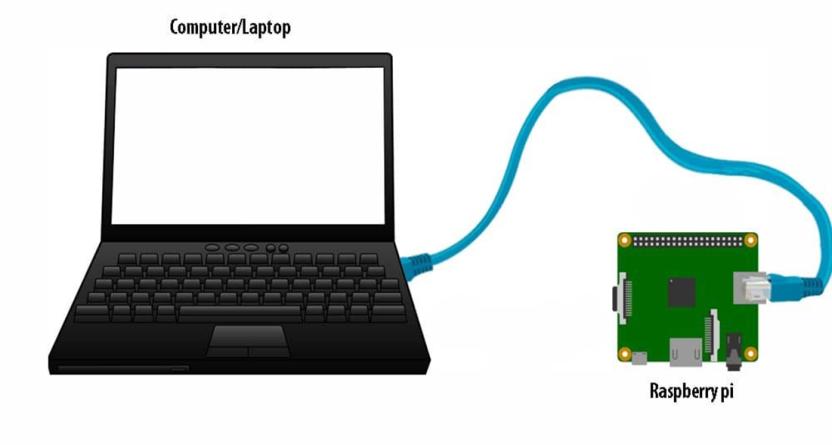


Figure 4.1 *Raspberry Pi interfacing*

As shown in the above picture, interfacing of our microcontroller unit is going to be done by PC connecting via LAN cable. So, there are various stages of interfacing such as downloading the Raspbian OS, then flash it into the SD card, after that we need to connect it to the putty and VNC viewer to see the screen of Raspbian OS.

**Pulse oximeter interfacing** 

It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices.



Figure 4.2 *Pulse oximeter*

This is also known as MAX30102. It operates on a single 1.8V power supply and 3.3V additional power supply for internal LEDs.•Here we have used the i2c communication protocol to communicate with MAX30102. I2C combines the best features of SPI and UARTs. With I2C, you can connect multiple slaves to a single master (like SPI) and you can have multiple masters controlling single, or multiple slaves.



Figure 4.3  *Pinout of pulse oximeter*

**Humidity Sensor interfacing**

Here we used a DHT11 sensor as a humidity sensor. It is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed). The temperature range of DHT11 is from 0 to 50 degrees Celsius with a 2-degree accuracy. The humidity range of this sensor is from 20 to 80% with 5% accuracy.

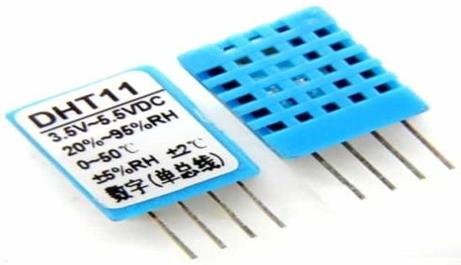


Figure 4.4 *DHT11*

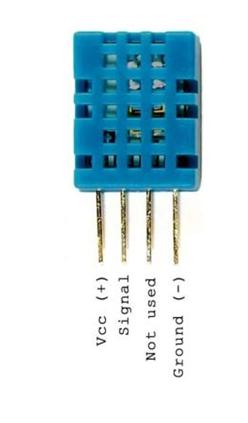


Figure 4.5 *Pinout of DHT11*

Above figure shows the pin configuration of the DHT11 humidity sensor.

**PIR sensor interfacing**

Here PIR states for **passive infrared** sensors. PIR sensors detect motion and are virtually always used to determine whether a human has entered or exited the sensor’s range. They are small, cheap, low power, simple to use and they don’t wear out. As a result, they’re often found in home and business equipment and gadgets. PIR motion sensors are also known as “Passive Infrared”, “Pyroelectric” or IR motion“ sensors.



Figure 4.6 *PIR sensor*

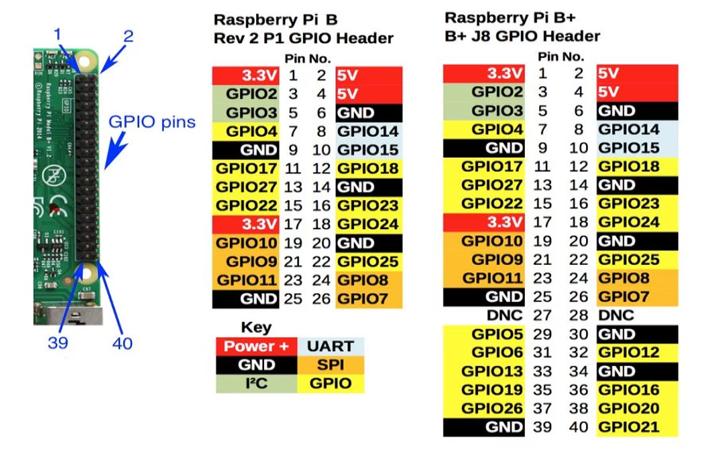


Figure 4.7 *Pinout of PIR sensor*

**GSM module interfacing with Raspberry Pi**

Here we used SIM868 GSM module to interface with raspberry pi. SIM868 is a handy, low power Raspberry Pi HAT which features multi communication functionalities: GSM, GPRS, GNSS and Bluetooth.

It allows your Pi to easily make a telephone call, send messages, connect to wireless Internet, global position, transfer data via Bluetooth, and so on.



Figure 4.8 *GSM module*

There are 2 types of interfacing for GSM modules but here we opted for USB interfacing over GPIO interfacing. In which we make use of a micro USB cable for connecting Raspberry Pi with the HAT module that turns in- turn provides the suitable power and establishes a serial communication.

**Servo Motor and Motor driver**

This little micro servo rotates 360 degrees fully forward or backwards, instead of moving to a single position.

We can use any servo code, hardware or library to control these servos. Good for making simple moving robots. Comes with five horns and attachment screws.

We here use an L298N DC motor driver who can control both speed and spinning direction of Motors. It can also control bipolar stepper motors.

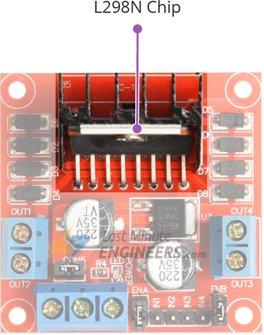


Figure 4.9 *Motor driver IC*

**USB Speaker interfacing**

We decided to add a speaker in this project for the calmness and entertainment of the baby. Whenever a baby cries, parents will get notified that their baby is crying. So, they can play some music through speakers from anywhere in the world using the webpage. So, that baby will be calm and stop crying.

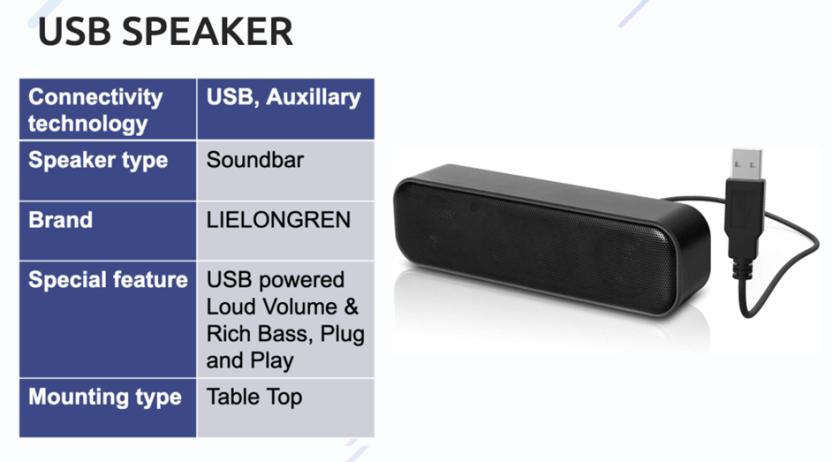
****

Figure 4.10 *USB speaker*

**Chapter V**

**Implementation and Test**

**Raspberry pi and PIR sensor**

As discussed,the PIR sensor is used to interface Raspberry pi. As soon as the PIR sensor senses the presence of a human, the sensor emits a 5V signal for one minute.It has a detection range of around 6–7 metres and is quite sensitive.When the PIR motion sensor detects a person, it sends a 5V signal to the Raspberry Pi's GPIO, and we use c code to describe what the Raspberry Pi should do when it detects a person.

* To run the code we need to type command as below.

gcc -Wall -o pirsensor pirsensor.c -lwiringPi

* To run the code we need to type command as:

sudo ./pirsensor

A picture containing text, electronics

Description automatically generated

Figure 5.1 *PIR sensor interfacing with Raspberry pi*

**Raspberry pi and Pulse - oximeter** MAX30102

The MAX30102 operates on a single 1.8V power supply  and a separate 3.3V power supply for the internal LEDs.Communication is through a standard I2C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times.Raspberry pi Pin 1 is connected to Vin of sensor,Pin 39 to GND of sensor,Pin 3 to SDA and Pin 5 to SCL of sensor.

Configure i2c on pi

* Command: cat /boot/config.txt | grep "i2c"
* Here, we can see that i2c is enabled now.
* ls /dev/ | grep i2c

Graphical user interface, text, application

Description automatically generated

Figure 5.2 *I2C interfacing enabled*

* Command: git clone <https://github.com/doug-burrell/max30102>
* This command will copy the github repository which has code for max30102 sensor.Run Arduino code on raspberry pi
* Go to this link and download windows zip file for arduino <https://www.arduino.cc/en/software>
* Download hardware board for converting arduino code into pi-executable code from here:
* <https://github.com/me-no-dev/RasPiArduino>
* Now go to C:/Programfilex86/Arduino/hardware/ and copy paste Piduno folder which you have downloaded from github.
* Next step is to download the toolchain for raspberry pi arm based binary on windows.
* [http://gnutoolchains.com/raspberry/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa0JJeUZoRXR2cElqVTFkSWh1c0QzRmlJd0s2QXxBQ3Jtc0trVWZxTWxlbE9zZkFFVS1lSjhORkl6ZC16VmM4NFROSE1pOE5wRUdoVVVsRzlsVWRqLUhJYnNBQmV3eEtrSzFiZF9TdjZmeUZZakhpLWRsZTVBb0NzVlo4RndDWWk5MlFoMmtCdFM5N25JcktRUTNWVQ&q=http%3A%2F%2Fgnutoolchains.com%2Fraspberry%2F)
* Goto link and install latest version. Leave directories as default, no change.Now open Arduino folder, go to hardware/piduino/platform.txt and change line number 5 as follows.This will be a path to cross toolchain for raspberry pi that we have already installed.Go to arduino IDE ->Tools-> board -> select raspberry pi.Open code of pulse oximeter.Click sketch->export compiled binary.It will take time to compile code.
* Run Arduino code on raspberry pi.Transport compiled binary to raspberry file through vnc viewer.Now, we need to give execution functionality to our binary file.Command: sudo chmod +x pulse.Execute program : sudo ./pulse ans see the results.

Text

Description automatically generated

Figure 5.3 *PIR sensor output on Raspberry pi*

**Raspberry pi and USB Camera**

We need to connect raspberry pi to putty and login to the device we use default username and password as stated earlier.When USB camera connected to RPi provides the USB devices list with the command lsusb.Here Microdia is the usb camera connected to raspberry pi.We need to install Guvcview and fswebcam application.To capture an image we can also go with the command on the terminal fswebcam image.jpg ,where it captures image and to display on PC with the command eog image.jpg followed by guvcview.

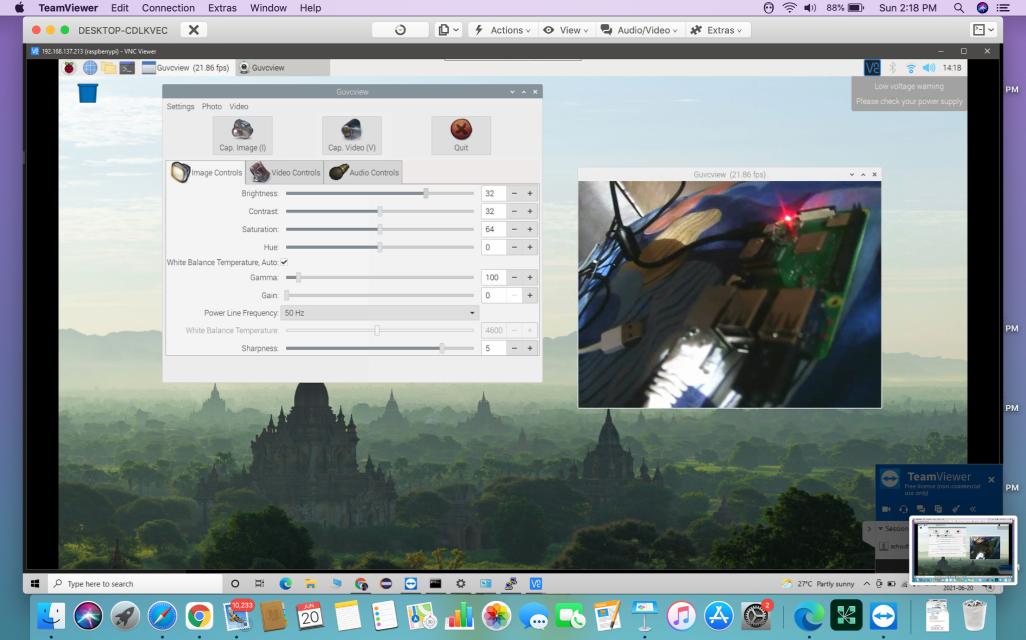


Figure 5.4 Video Capture by Guvcview

To stream live video install motion.cd /etc/motion/ and sudo nano motion.conf - scroll down and change ‘stream-localhost on’ to ‘stream-local host off’ and press ctrl-x and y and ente,then enter sudo motion -n and ifconfig - ethernet address 193.168.137.213 followed by :8081.

**Raspberry pi and GSM Module**

* SIM 868 module is being used in this device which is a complete Quad-Band GSM/GPRS module which combines GNSS technology for satellite navigation. In the left side of the module you can see a yellow jumper connected to the suitable pins. There are four pairs of pins in which the jumpers must be shorted to make three terminal pairs (A,B,C).
* A: control the SIM868 through USB TO UART
* B: control the SIM868 through Raspberry Pi
* C: access Raspberry Pi through USB TO UART
* Here we make use of A terminal, i.e. the jumpers must be connected vertically in the first two pins as shown in the figure.

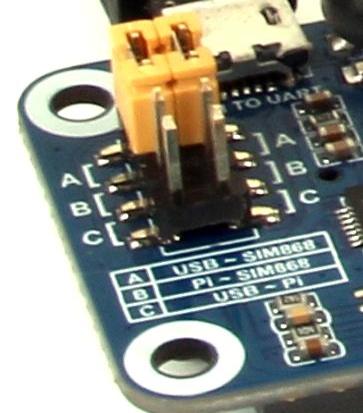


Figure 5.5 SIM868 interfacing Raspberry pi

* Insert a sim card to the HAT module to perform GSM operations.After connecting the USB, the power led will be in an on state. Press the PWRKEY for 3 seconds and remove it. The STA LED (status) will also be in one state.The NET led will blink in a continuous manner and after obtaining the range for sim card, the blinking rate will be reduced.The "AT" or "at" or “aT” or “At” prefix must be set at the beginning of each Command line. To terminate a Command line enter <CR>. Commands are usually followed by a response that includes. "<CR><LF><response><CR><LF>" Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.

**Raspberry pi and Servo motor**

* Connect the VCC and GND of the Tower Pro SG90 Servo Motor to +5V and GND pins of the power supply. Then connect the PWM Pin of the Servo Motor to Physical Pin 22 of Raspberry Pi i.e, GPIO 25.Make the ground common between Raspberry Pi and the Power Supply of the Servo Motor.

Diagram, schematic

Description automatically generated

Figure 5.6 Servo motor and L298N interfacing Raspberry pi

Commands for linux

* gpio -g mode 18 pwm
* gpio pwm-ms
* gpio pwmc 192
* gpio pwmr 2000
* gpio -g pwm 18 150
* gpio -g pwm 18 200

**Raspberry pi and Humidity & Temperature sensor-DHT11** 

* Connect the DHT11 Sensor Signal to Raspberry Pi PIN 7 (GPIO PIN 4) andDHT11 Sensor Vcc+ to Raspberry Pi 5V(Pin 2),DHT11 Sensor GND to Raspberry Pi GND(Pin 6).

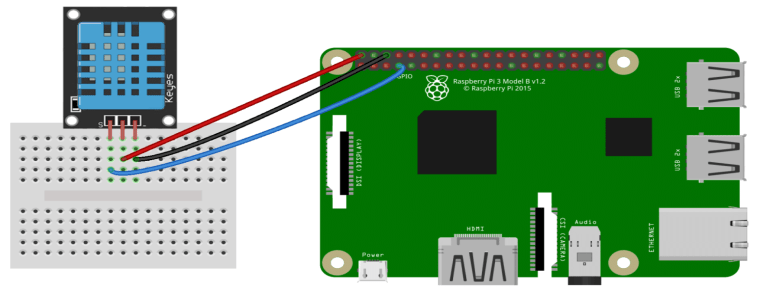


Figure 5.7 DHT11 interfacing Raspberry pi

* The program can be compiled with this gcc command
* gcc -o dht11 dht11.c -lwiringPi -lwiringPiDev
* By using this command on the SSH terminal we can see the output of humidity and temperature (in °C and °F) readings
* sudo ./dht11

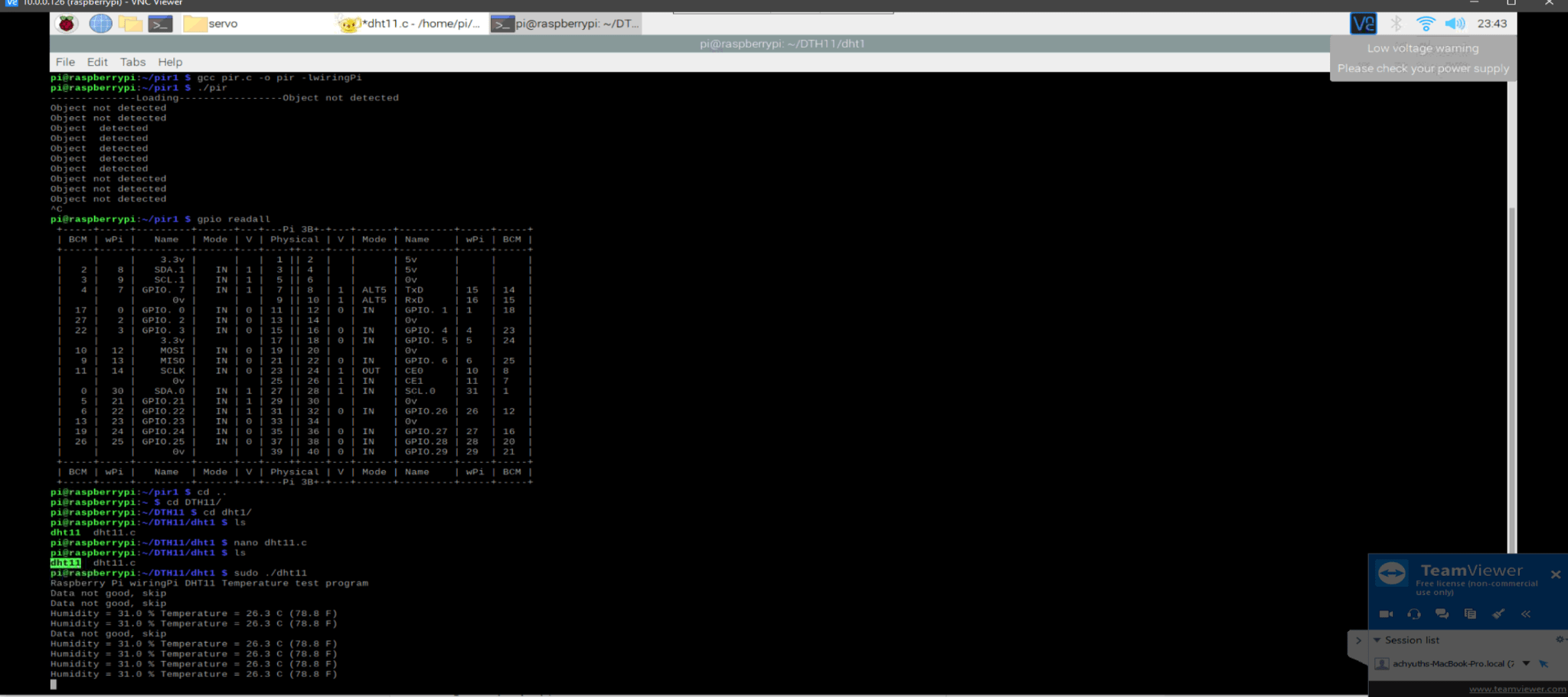


Figure 5.8 Output of DHT11 interfacing Raspberry pi

**Raspberry pi and USB Speaker**

‌Start the putty and we will see the IP address and enter the address.After that you will see the login prompt.The default login for Raspberry Pi OS is pi with the password raspberryLog into VNC viewer with the default username is “pi” and password is “raspberry”.‌Click on to the speaker icon and select USB2.0 Device.‌When USB speaker connected to RPi provides the USB devices list with the command:lsusb.Here GEMBIRD is the usb speaker connected to raspberry pi.

Making a directory as speaker and changing into that directory.Creating a text editor as nano speaker.c.Using the HTML code we can see the output in the webpage.

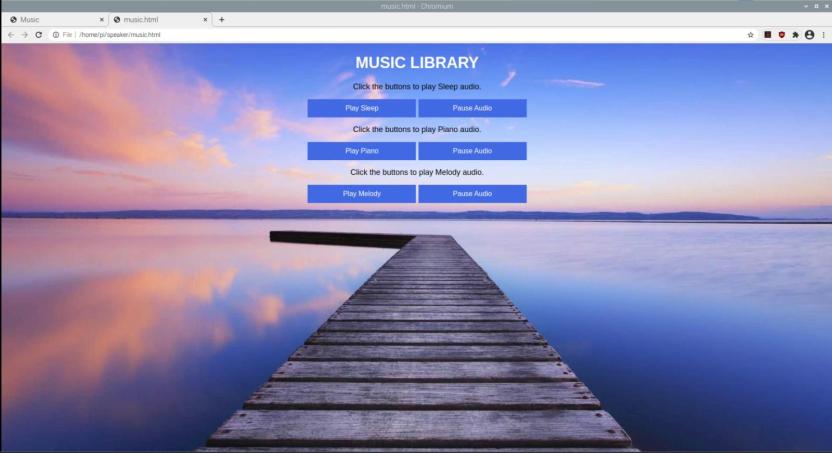


Figure 5.9 Output of USB speaker interfacing Raspberry pi

**Raspberry pi and Web Server**

* We will configure the RPi to be a web server that uses various server-side scripting techniques to display sensor data.Custom C/C++ code is described that can push sensor data to the Internet.
* An RPi that is connected to a sensor and running a web server can be used to present information to the web when it is requested to do so by a web browser. Communications take place using the Hypertext Transfer Protocol (HTTP).
* An RPi can initiate contact with a web server using HTTP requests to send and receive data. A C/C++ program is written that uses TCP sockets to build a basic web browser, which can communicate over HTTP, or if necessary, securely over secure HTTP (HTTPS).A custom C++ client and server are presented that can intercommunicate at high speeds with a user-defined communications protocol.
* Code is written to enable the RPi to use HTTP and MQTT to send data to, and receive data from, web services such as ThingSpeak and IBM Bluemix IoT. This code enables you to build large arrays of sensors that can intercommunicate and store data on remote servers. In addition, these web services can be used to visualize the data that is stored.
* The Nginx server is currently available through the Raspbian distribution. You can use the following commands to install it:
* pi@erpi ~ $ **sudo apt update**
* pi@erpi ~ $ **sudo apt install nginx**
* pi@erpi ~ $ **sudo reboot**

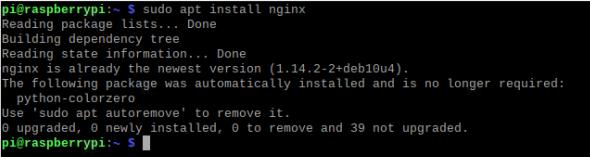


Figure 5.10 Install nginx

* To create a simple web page for the RPi web server, you can use the nano editor and some basic HTML syntax as follows:
* pi@erpi /var/www/html $ **sudo nano index.html**
* pi@erpi /var/www/html $ **more index.html**
* <HTML><TITLE>RPi First Web Page</TITLE>
* <BODY><H1>RPi First Page</H1>
* The Raspberry Pi test web page.
* </BODY></HTML>

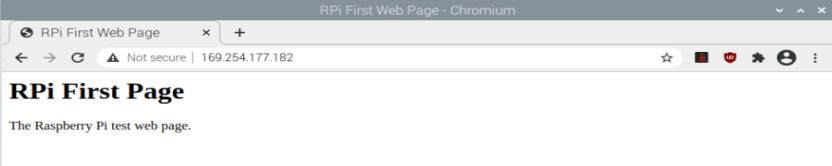


Figure 5.11 RPi Web page

**Real time operation**

A real-time operation is an operation designed to support real-time applications that process data in real time, without buffer delays. Processing time requirements are measured in tenths of seconds or smaller time increments (including any OS delay).A time-bound system with well-defined, definite time restrictions is known as a real-time system.The system will fail if processing is not done within the given limits. They are either time-sharing or event-driven.

There are various codes we need to execute for the real time operation of various components.

for example real time operation code for Pulse Oximeter is as below,

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include<fcntl.h>

#include<unistd.h>

#include<termios.h> // using the termios.h library

#include<string>

#include<iostream>

#include <wiringPi.h>

#define NUM\_THREADS 2

/\* create thread argument struct for thr\_func() \*/

typedef struct \_thread\_data\_t {

int tid;

double stuff;

} thread\_data\_t;

/\* thread function \*/

void \*thr\_func(void \*arg) {

if(wiringPiSetup() == -1){

printf("setup wiringPi failed !");

exit(1);

}

pinMode( 29, OUTPUT );

digitalWrite( 29, HIGH );

sleep(1);

unsigned char hr[100];

int file, count;

if ((file = open("/dev/ttyAMA0", O\_RDWR | O\_NOCTTY | O\_NDELAY))<0) {

perror("UART: Failed to open the file.\n");

pthread\_exit(NULL);

}

struct termios options; // the termios structure is vital

tcgetattr(file, &options); // sets the parameters for the file

// Set up the communications options:

// 115200 baud, 8-bit, enable receiver, no modem control lines

options.c\_cflag = B115200 | CS8 | CREAD | CLOCAL;

options.c\_iflag = IGNPAR | ICRNL; // ignore parity errors

tcflush(file, TCIFLUSH); // discard file information

tcsetattr(file, TCSANOW, &options); // changes occur immmediately

//unsigned char transmit[20] = "Hello Raspberry Pi!"; // send string

//if ((count = write(file, &transmit, 20))<0){ // transmit

//perror("Failed to write to the output\n");

//return -1;

//}

sleep(29); // give the Arduino a chance to respond

unsigned char receive[100]; // declare a buffer for receiving data

printf("<div><h3>Give 30s to respond:/ Thank you for your paitence</div></h3>\n");

if ((count = read(file, (void\*)receive, 100))<0){ //receive data

perror("Failed to read from the input\n");

pthread\_exit(NULL);

}

else if (count==0) printf("There was no data available to read!\n");

else printf("<div><h3>The following was read in [%d]: %s</div></h3></para></html>\n",count,receive);

close(file);

//unsigned char hr = receive[2-4];

//printf("<div><h3>Heartrate : %s</div></h3>\n",hr[2-4]);

//printf("%s",hr);

digitalWrite( 29, LOW );

pthread\_exit(NULL);

}

int main(int argc, char \*\*argv) {

int policy = SCHED\_FIFO;

pthread\_t tid;

pthread\_attr\_t tattr;

int ret;

int newprio = 1;

struct sched\_param param;

ret = pthread\_attr\_init (&tattr);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 1 failed \n ");

return 1;}

ret = pthread\_attr\_getschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 3 failed \n");

return 1;}

param.sched\_priority = newprio;

ret = pthread\_attr\_setschedpolicy(&tattr, policy);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 5 failed \n");

return 1;}

ret = pthread\_attr\_setschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 7 failed \n");

}

int i, rc;

/\* create a thread\_data\_t argument array \*/

while (1) {

/\* create threads \*/

if ((ret = pthread\_create(&tid, &tattr, thr\_func, NULL))) {

fprintf(stderr, "error: pthread\_create, ret: %d\n", ret);

return EXIT\_FAILURE;

}

printf("-------------------------------The End---------------------------\n");

pthread\_join(tid, NULL);

}

return EXIT\_SUCCESS;

}

Code for Humidity sensor and PIR sensor is as below,

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <wiringPi.h>

#include <stdint.h>

#define MAXTIMINGS 85

#define DHTPIN 7

#define PIRPin 25

int dht11\_dat[5] = { 0, 0, 0, 0, 0 };

/\* create thread argument struct for thr\_func() \*/

typedef struct \_thread\_data\_t {

int tid;

double stuff;

} thread\_data\_t;

/\* thread function \*/

void \*dht11(void \*arg) {

uint8\_t laststate = HIGH;

uint8\_t counter = 0;

uint8\_t j = 0, i;

float f;

dht11\_dat[0] = dht11\_dat[1] = dht11\_dat[2] = dht11\_dat[3] = dht11\_dat[4] = 0;

wiringPiSetup () ;

pinMode( DHTPIN, OUTPUT );

digitalWrite( DHTPIN, LOW );

delay( 18 );

digitalWrite( DHTPIN, HIGH );

delayMicroseconds( 40 );

pinMode( DHTPIN, INPUT );

for ( i = 0; i < MAXTIMINGS; i++ )

{

counter = 0;

while ( digitalRead( DHTPIN ) == laststate )

{

counter++;

delayMicroseconds( 1 );

if ( counter == 255 )

{

break;

}

}

laststate = digitalRead( DHTPIN );

if ( counter == 255 )

break;

if ( (i >= 4) && (i % 2 == 0) )

{

dht11\_dat[j / 8] <<= 1;

if ( counter > 50 )

dht11\_dat[j / 8] |= 1;

j++;

}

}

if ( (j >= 40) &&

(dht11\_dat[4] == ( (dht11\_dat[0] + dht11\_dat[1] + dht11\_dat[2] + dht11\_dat[3]) & 0xFF) ) )

{

f = dht11\_dat[2] \* 9. / 5. + 32;

printf( "<div><h3>Humidity = %d.%d %% Temperature = %d.%d C (%.1f F)</h3></div>\n",

dht11\_dat[0], dht11\_dat[1], dht11\_dat[2], dht11\_dat[3], f );

}else {

printf( "Data not good, skip\n" );

delay(10);

}

pthread\_exit(NULL);

}

void \*pir(void \*arg) {

wiringPiSetup () ;

int t=1;

//char command[] = "clear";

if(wiringPiSetup() == -1){

printf("setup wiringPi failed !");

exit(1);

}

pinMode(PIRPin,INPUT);

if(!(digitalRead(PIRPin))){

printf("<div><h3>Motion not detected</div></h3>\n");

}

else{

printf("<div><h3>Someone is around baby</div></h3>\n");

}

pthread\_exit(NULL);

}

int main(int argc, char \*\*argv) {

int policy = SCHED\_FIFO;

pthread\_t tid,tid1;

pthread\_attr\_t tattr,tattr1;

int ret,ret1;

int newprio = 1;

int newprio1 = 15;

struct sched\_param param,param1;

ret = pthread\_attr\_init (&tattr);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 1 failed \n ");

return 1;}

ret1 = pthread\_attr\_init (&tattr1);

if(ret1 != 0) {

printf("pthread\_attr\_setinheritsched() 2 failed \n");

return 1;}

ret = pthread\_attr\_getschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 3 failed \n");

return 1;}

ret1 = pthread\_attr\_getschedparam (&tattr1, &param1);

if(ret1 != 0) {

printf("pthread\_attr\_setinheritsched() 4 failed \n");

return 1;}

param.sched\_priority = newprio;

param1.sched\_priority = newprio1;

ret = pthread\_attr\_setschedpolicy(&tattr, policy);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 5 failed \n");

return 1;}

ret = pthread\_attr\_setschedpolicy(&tattr1, policy);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 6 failed \n");

return 1;}

ret = pthread\_attr\_setschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 7 failed \n");

return 1;}

ret1 = pthread\_attr\_setschedparam (&tattr1, &param1);

if(ret1 != 0) {

printf("pthread\_attr\_setinheritsched() 8 failed \n");

return 1;}

int i, rc;

/\* create a thread\_data\_t argument array \*/

while (1) {

/\* create threads \*/

if ((ret = pthread\_create(&tid, &tattr, dht11, NULL))) {

fprintf(stderr, "error: pthread\_create, ret: %d\n", ret);

return EXIT\_FAILURE;

}

printf("priority=%d",param1.sched\_priority);

if ((ret1 = pthread\_create(&tid1, &tattr1, pir, NULL))) {

fprintf(stderr, "error: pthread\_create, ret: %d\n", ret);

return EXIT\_FAILURE;

}

pthread\_join(tid, NULL);

pthread\_join(tid1, NULL);

printf("-------------------------------The End---------------------------\n");

}

return EXIT\_SUCCESS;

}

and at the last code for servomotor is as below,

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include<fcntl.h>

#include<unistd.h>

#include<termios.h> // using the termios.h library

#include <wiringPi.h>

#define NUM\_THREADS 2

/\* create thread argument struct for thr\_func() \*/

typedef struct \_thread\_data\_t {

int tid;

double stuff;

} thread\_data\_t;

/\* thread function \*/

void \*thr\_func(void \*arg) {

int t;

printf ("Raspberry Pi wiringPi test program\n");

wiringPiSetupGpio();

pinMode (18, PWM\_OUTPUT) ;

pwmSetMode (PWM\_MODE\_MS);

pwmSetRange (2000);

pwmSetClock (192);

for(t=0;t<=5;t++){

pwmWrite(18,1);

usleep(100000);

pwmWrite(18,210);

usleep(100000);

}

pwmWrite(18,0);

pthread\_exit(NULL);

}

int main(int argc, char \*\*argv) {

int policy = SCHED\_FIFO;

pthread\_t tid;

pthread\_attr\_t tattr;

int ret;

int newprio = 1;

struct sched\_param param;

ret = pthread\_attr\_init (&tattr);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 1 failed \n ");

return 1;}

ret = pthread\_attr\_getschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 3 failed \n");

return 1;}

param.sched\_priority = newprio;

ret = pthread\_attr\_setschedpolicy(&tattr, policy);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 5 failed \n");

return 1;}

ret = pthread\_attr\_setschedparam (&tattr, &param);

if(ret != 0) {

printf("pthread\_attr\_setinheritsched() 7 failed \n");

}

int i, rc;

/\* create a thread\_data\_t argument array \*/

/\* create threads \*/

if ((ret = pthread\_create(&tid, &tattr, thr\_func, NULL))) {

fprintf(stderr, "error: pthread\_create, ret: %d\n", ret);

return EXIT\_FAILURE;

}

printf("-------------------------------The End---------------------------\n");

pthread\_join(tid, NULL);

return EXIT\_SUCCESS;

}

**Final Prototype**

After implementing all the components and codes our prototype looks like below images:

A picture containing indoor, floor, wood, hard

Description automatically generated

A picture containing floor

Description automatically generatedA picture containing diagram

Description automatically generated

**Chapter VI**

**Evaluation**

**Introduction**

In this chapter we would like to discuss about the final completion of the project and also inform the user about the minimum requirements for the setup of this product. Sometimes it is hard to solve any issue occurred in new product. So, here we also include the possible issues and solution of it under the topic of troubleshooting.

**Minimum Requirements**

Users should have a nice internet connectivity for both system and streaming to stream all the data without any buffer delay and with accuracy. They also need local storage for storing live streaming systems. On top of that, they should have a laptop or mobile with internet connectivity to view and operate websites. At last they should have a conventional cradle to attach this system with.

**Troubleshooting**

1. Faced problem during connection using wiring pi library because it use different pin configuration.

Solution: Then use command gpio readall to get perfect pin configuration.

1. Faced problem using the DHT11 sensor was damaged during testing of high temperature.

Solution: Then you need to purchase a new DHT11 sensor.

1. Faced problem connecting VNC viewer.

Solution: Every time raspberry pi was allocated different IP addresses. Then use raspberrypi.local as a server name in VNC viewer to solve the problem.

1. Faced problem using nginx to publish data and website output was not displayed in web browser.

Solution: port 80 was in already use for motion service to display video streaming output, to solve this we can forward motion service output in port 8081 and nginx service output in 80 port.

**Chapter VII**

**Conclusion**

The aim of this project was to develop a smart baby cradle where parents can monitor their baby.We have the features like wetness detection,cry detection,video monitoring,play music and swing the cradle when baby cries,SIDS monitor and also send alerts during emergencies.

For better experience we have made a webpage where the user can login to the website and monitor the baby activities and also if there is no internet the user will get the emergency alert through message by the GSM module feature.We have made a bandaid which can be placed on the baby feet to detect the pulse rate as the baby skin is very sensitive.

The product is designed in a plug and play mode.We have used Raspberry pi to interface all these sensors and created a website to be user friendly.Moreover this product helps the parent to take some rest and save time by monitoring their baby in real time.

**Future Work**

1. We will create a global cloud to store the video footage of the baby.
2. We will add an option in the webpage to upload the user's desired music. So, they can play the music according to their likes and dislikes.
3. At the last but not least we will also provide the facility to change the mobile number associated with system to send the emergency notifications.

**Chapter VIII**

**User’s Guide**

**Powering Up**

Users need two usb cables to power up. First one is a mini USB cable to power raspberry pi and another one to power an arduino.

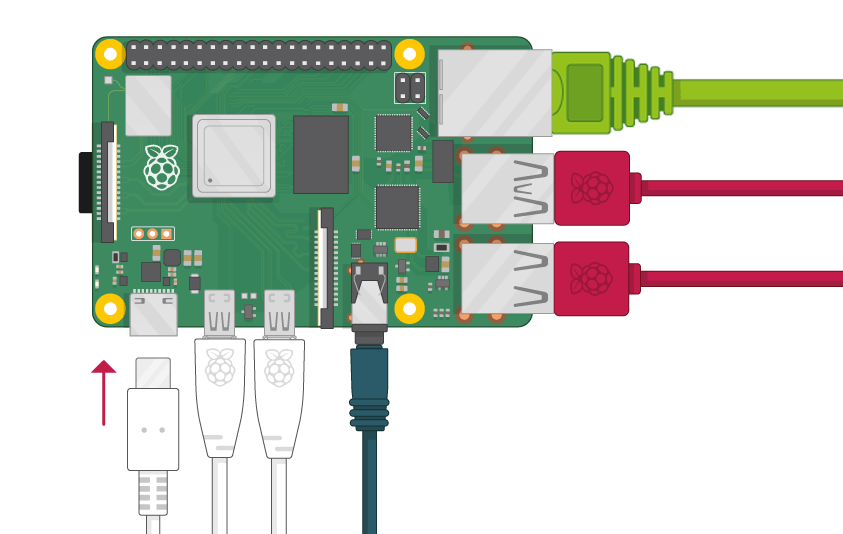


Figure 9.1 *Powered DLP 2000 EVM and Beaglebone Black*

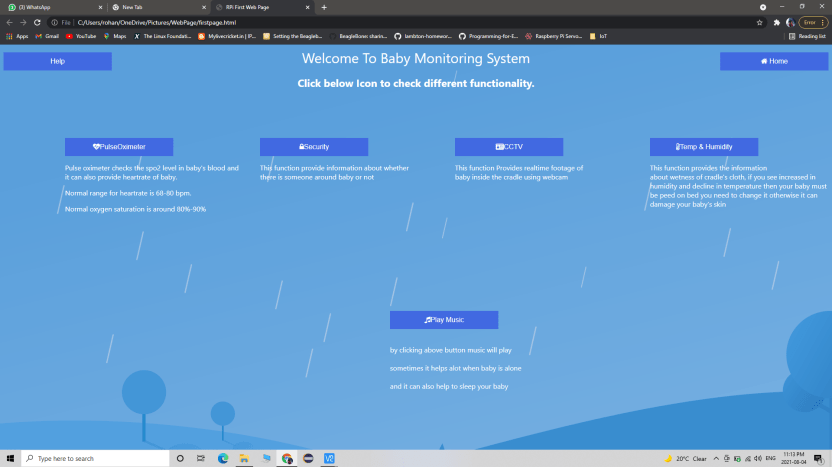
If both the modules are successfully powered up, you will notice a heartbeat blinking LED on Raspberry pi and green led will on arduino. Also check if GSM module led is on, if it is on then press GSM power key for 3s until another red led on gsm blinks.

**Connecting Wi-Fi** 

Through the desktop, connect to WiFi. For Windows users, there is a wifi symbol on the right bottom of the navigation bar. For linux users there is a wifi symbol on the top right corner.

Open the browser and type the IP address of your router.

Then, website homepage will appear like this:



There are many buttons available for different functionality.

1. Pulse Oximeter:

When you click this button, it will provide you with approximate data of heart rate and oxygen level of the baby. User needs to attach the pulse oximeter in the baby's leg correctly to have correct data. Normal heart rate is 80-130 and spo2 level should be 95%-100%. When readings go above, the system will notify parents by website and through registered mobile numbers also.

1. Security:

This button will provide PIR sensor data, if the PIR sensor detects human or animal it will print a message that “some one is around the baby”.

1. CCTV:

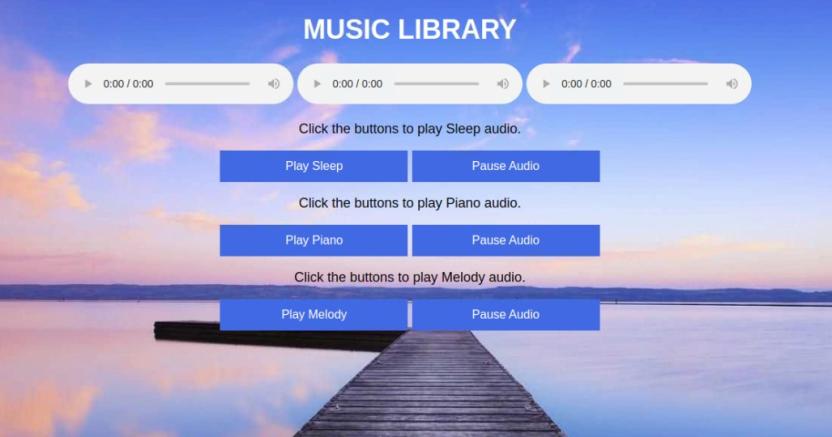
This button provides live footage of the baby inside the cradle so parents can monitor whenever they want.

1. Temp & Humidity:

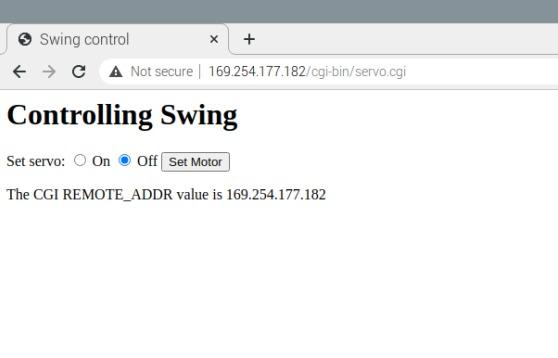
Surrounding conditions are important to know and it's super important to act based on this information. This button provides data of the temperature and humidity sensor that is attached to the cradle. If a baby urinated, parents will know it by increasing the humidity value. In case of fire, the temperature will increase and the website will notify.

1. Play Music:

This button allows you to play music through a website. Users can increase or decrease volume also.



1. swing control:

It will provide the facility of swing control of the baby. When button is pressed, it will ask the user to choose between on or off swing, when user select on and pressed set motor, motor will rotate and swing action is performed through servo motor.

**Chapter IX**

**References**

1.WoutersPF,Gehring H,Meyfroidt G,Ponz L,Gil-Rodriguez J,Hornberger C,et al.Accuracy of pulse oximeters: the European multi-center trial. Anesth Analg. 2002;**94**(1 Suppl):S13–6.

2.I.Jhun,D.A.Mata,F.Nordio,M.Lee,J.Schwartz, andA.Zanobetti,‘‘Ambient temperature and sudden infant death syndrome in the United States,’’ Epidemiology, vol. 28, no. 5, pp. 728–734, 2017.

3.B. J. Taylor, J. Garstang, A. Engelberts, T. Obonai, A. Cote, J. Freemantle, M. Vennemann, M. Healey, P. Sidebotham, E. A. Mitchell, and R. Y. Moon, ‘‘International comparison of sudden unexpected death in infancy rates using a newly proposed set of cause-of-death codes,’’ Arch. Disease Childhood, vol. 100, no. 11, pp. 1018–1023, 2015.

4.ADA fruit. “ADA fruit.” ADA fruit, 2021, https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor. Accessed 28 06 2021.

5.Maker pro. “Maker pro.” Maker pro, 2021, https://maker.pro/raspberry-pi/tutorial/how-to-interface-a-pir-motion-sensor-with-raspberry-pi-gpio. Accessed 29 06 2021.

6.Doug borrel, June 9,2020, MAX30102 Pulse Oximetry Sensor code for Raspberry Pi, <https://github.com/doug-burrell/max30102>

7. Scott Campbell,BASICS OF THE I2C COMMUNICATION PROTOCOL,  <https://www.smart-prototyping.com/Pulse-Oximeter-and-Heart-Rate-Sensor-MAX30102>

9.Dmitry Grigoryev, Oct 20, 20202,How can Rpi read the MAX30100 / MAX30102 Oximeter?, <https://raspberrypi.stackexchange.com/questions/111955/how-can-rpi-read-the-max30100-max30102-oximeter>

10.Vivek Gite, March 14, 2018,Raspberry PI 3 model B+ Released: Complete specs and pricing,<https://www.cyberciti.biz/hardware/raspberry-pi-3-model-b-released-specs-pricing/>

11.Batz, V. M. (2019, January 4). Introducing PiCockpit and PiDoctor – how to test your Raspberry Pi and get information about it. Pi3g.com. <https://pi3g.com/2019/01/05/introducing-picockpit-and-pidoctor-how-to-test-your-raspberry-pi-and-get-information-about-it>.

12.amazon. (n.d.). <https://www.amazon.ca/dp/B07Z2GZTYY/ref=redir_mobile_desktop?_encoding=UTF8&psc=1&ref=ppx_pop_mob_b_asin_title>

13.How to Make Raspberry Pi Webcam Server and Stream Live Video || Motion + Webcam + Raspberry Pi. (n.d.). instructables circuits. <https://www.instructables.com/How-to-Make-Raspberry-Pi-Webcam-Server-and-Stream-/>

14.install GTK+ UVC Viewer (UNOFFICIAL) on Raspberry Pi. (n.d.). snapcraft. <https://snapcraft.io/install/guvcview/raspbian>

15.Virtual Network Computing. (n.d.). wikipedia. <https://en.wikipedia.org/wiki/Virtual_Network_Computing>

16.DEREK MOLLY. (2016). Exploring Raspberry pi. WILEY. <https://www.amazon.ca/Exploring-Raspberry-Pi-Interfacing-Embedded/dp/1119188687>

17.Motion Guide. (n.d.). <https://motion-project.github.io/3.4.1/motion_guide.html>

18.waveshare. (n.d.). GSM/GPRS/GNSS/Bluetooth HAT for Raspberry Pi. Waveshare. Retrived from - <https://www.waveshare.com/gsm-gprs-gnss-hat.htm>.

19.Mahesh. (2018, May 2). Raspberry Pi: rhydoLABZ-wiki. rhydoLABZwiki. Retrived from - <https://www.rhydolabz.com/wiki/?cat=39>.

20.SIM868 GSM/GPRS/GNSS/Bluetooth HAT for Raspberry Pi. DIYElectronics. (n.d.). <https://www.diyelectronics.co.za/store/hats/1800-sim868-gsmgprsgnss-hat-for-raspberry-pi.html>.

21.Check service provider frequency  (n.d.) [https://www.frequencycheck.com](about:blank)

22.Interface L298N DC Motor Driver Module with Arduino, Retrived from

<https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/>

23.Electronics Hub,February 14, 2018, Raspberry Pi Servo Motor Interface, Retrieved from <https://www.electronicshub.org/raspberry-pi-servo-motor-interface-tutorial/>

24.Jim blom, May 8,2019, Raspberry pi GPIO, retrived from <https://learn.sparkfun.com/tutorials/raspberry-gpio/c-wiringpi-api>

25.scott campbell. (2018). HOW TO SET UP THE DHT11 HUMIDITY SENSOR ON THE RASPBERRY PI. circuit basics. <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/>

26.DHT11 Humidity & Temperature Sensor. (2017). Mouser Electronics. <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>

‌27.USB Computer Speaker, Laptop Speaker with Stereo Sound & Enhanced Bass, Portable Mini Sound Bar for Windows PCs, Desktop Computer and Laptops. (n.d.). Amazon.<https://www.amazon.ca/dp/B086JXJ1LF/ref=redir_mobile_desktop?_encoding=UTF8&psc=1&ref=ppx_pop_mob_b_asin_title>

28.MDN Web Docs. (n.d.). <audio>: The Embed Audio element.<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio>

29.Derek Molloy, (2016). Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux, New York: Wiley.

30.Stackshare, Azure websites vs NGINX, retrieved from: https://stackshare.io/stackups/azure-websites-vs-nginx

31.Setting up an NGINX web server on a Raspberry Pi, Retrieved from: <https://www.raspberrypi.org/documentation/remote-access/web-server/nginx.md>