**Smart Baby Monitoring System**

**by**

***Samudrala Chandra Pranav 21BPS1535***

***Ch Naga Sai Abheeshek Varma 21BPS1509***

***Gangadi Varshith Reddy 21BPS1511***

***Bellamkonda Raghu 21BPS1386***

***Under the guidance of,***

**Dr. Sridhar C**

Text, logo

Description automatically generated

**School Of Computer Science and Engineering**

**Vellore Institute of technology, Chennai-600127**

**April-2024**

**DECLARATION**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Samudrala Chandra Pranav | Ch Naga Sai Abheeshek Varma |  |
| 21BPS1535 | 21BPS1509 |  |

|  |  |
| --- | --- |
|  |  |
| Gangadi Varshith Reddy | Bellamkonda Raghu |
| 21BPS1511 | 21BPS1386 |
|  |  |

**CERTIFICATE**

It is certified that the work contained in the Continuous Assessment and Mini project(CAMP) titled “**Smart Baby Monitoring System**,” by “Abheeshek, Raghu, Varshith and Pranav, bearing Roll No:21BPS1509, 21BPS1386, 21BPS1511, 21BPS1535” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

**Signature of Supervisor**

**Dr. Sridhar C**

**SENSE**

**VIT, Chennai**

**April 2024**

**ACKNOWLEDGEMENTS**

I would like to extend my sincerest gratitude to the Dean of Academics for their unwavering support throughout my academic journey. Their guidance and leadership have been instrumental in shaping my educational experience, and I am truly grateful for the opportunities they have provided me with to excel and grow both academically and personally.I am also deeply grateful to my Head of Department for their invaluable mentorship and support. Their dedication to fostering a stimulating learning environment and their commitment to academic excellence have inspired me to push the boundaries of my knowledge and capabilities. Their encouragement and guidance have been instrumental in helping me navigate the challenges of my academic pursuits.To each of these individuals, I extend my heartfelt appreciation for their role in my academic journey. Their support and guidance have been instrumental in shaping my growth and development, and I am truly grateful for the impact they have had on my education and professional aspirations.

**LIST OF TABLES**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Table 1 : Component table | 8 |
| **Component table**   |  |  | | --- | --- | |  | Arduino | |  | Rain Drop Sensor | |  | LDR Sensor | |  | Laser Module | |  | ESP32-CAM | |  | Buzzer | |  | Sound Sensor | |  | DHT Sensor | |  | Servo Motor | |  |

**LIST OF ABBREVIATION**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| API | Application Programming Interface |
| UI | User Interface |
| ES | Embedded Systems |
| LDR | Light Dependent Resistor |
| APP | Software application |

**Abstract**

This report presents the design and development of a Smart Baby Monitoring System leveraging the capabilities of Arduino and an array of sensors. The system prioritizes the safety and comfort of infants by providing real-time environmental monitoring. A DHT sensor continuously tracks temperature and humidity levels within the crib. If the temperature exceeds a predefined threshold, the system automatically activates a fan to regulate the environment. Addressing the emotional well-being of the baby, an integrated sound sensor detects crying episodes. Upon detection, the system triggers a servo motor to gently swing the cradle, promoting a soothing effect. To safeguard against potential rollovers, a unique combination of a laser and an LDR sensor monitors movement within the crib. If the laser beam loses contact with the LDR sensor, indicating movement away from the designated area, the ESP32 camera captures a real-time photo. This photo is then promptly sent to the user's smartphone through a custom-built Telegram bot notification, enabling immediate action. The system also boasts a user-friendly Node-RED dashboard that displays real-time data on temperature, humidity, and sound levels, providing parents with a comprehensive overview of their baby's environment. This report delves into the functionalities of the Smart Baby Monitoring System, analyses its effectiveness in ensuring infant safety and fostering comfort, and identifies potential areas for further development, paving the way for future advancements in baby monitoring technology.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | | Page No |
| Title | | 1 |
| Declaration | | 2 |
| Certificate | | 3 |
| Acknowledgements | | 4 |
| List of Tables | | 5 |
| List of Symbols and Abbreviations | | 7 |
| Abstract | | 8 |
| Table of Contents | | 9 |
| Contents | |  |
| 1 | Introduction | 10 |
| 2 | Literature Review | 11 |
| 3 | Hardware Description | 13 |
| 4 | Application Specification | 14 |
|  | 4.1 System Functionalities | 14 |
|  | 4.2 User Interface | 14 |
| 5 | Experimental Procedure | 16 |
|  | 5.1 Hardware Setup | 16 |
|  | 5.2 Software development | 16 |
|  | **5.3. System Testing and Calibration** | 46 |
|  | **5.4. Conclusion** | 47 |
| 6 | Results and Discussion | 49 |
| 9 | Conclusions and Future Scope | 51 |
|  | References | 53 |
|  | Appendix | 54 |

**INTRODUCTION**

The well-being of a newborn is a top priority for parents, and ensuring a safe and comfortable environment is crucial for healthy development. Traditional baby monitoring methods often rely on visual checks, basic audio monitors, or rudimentary moisture detection pads, which can be time-consuming, lack real-time data, or be prone to inaccuracies. This report presents the design and implementation of a **Smart Baby Monitoring System** that addresses these limitations by leveraging modern technology to provide a comprehensive and proactive solution.

This innovative system utilizes an **Arduino microcontroller** as the central processing unit, enabling the integration of various sensors to create a comprehensive monitoring solution. The core functionalities of the system can be summarized as follows:

* **Environmental Monitoring:**
  + **Temperature and Humidity:** A **DHT sensor** continuously monitors the temperature and humidity within the crib. This real-time data allows for prompt intervention if the environment becomes too hot, humid, or cold, potentially causing discomfort for the baby. The system can be programmed to trigger a **fan** to regulate the temperature or a **dehumidifier** (if applicable) to adjust humidity levels.
  + **Dampness Detection:** A dedicated **moisture sensor** is strategically placed within the crib or on the mattress pad to detect dampness. This early warning system can alert parents to potential issues like diaper leaks or excessive perspiration, allowing them to address the situation before it becomes a source of discomfort or irritation for the baby.
* **Cry Detection and Soothing Mechanism:** An integrated **sound sensor** detects crying episodes. Upon detection, the system activates a **servo motor** gently to swing the cradle. This automated soothing mechanism can promote calmness and help the baby fall back asleep.
* **Anti-Rollover Detection and Notification:** A unique combination of a **laser** and an **LDR sensor** safeguards against potential rollovers. The laser beam continuously shines on the LDR sensor. If the baby moves and the laser loses contact with the LDR sensor, indicating movement away from the designated area, the system triggers an **ESP32 camera** to capture a real-time photo. This photo is then promptly sent to the user's smartphone through a custom-built **Telegram bot notification**. This immediate alert allows parents to take action if necessary.

Furthermore, the system boasts a user-friendly **Node-RED dashboard**. This dashboard displays real-time data on temperature, humidity, sound levels, and dampness, providing parents with a comprehensive overview of their baby's environment. This information empowers parents to make informed decisions about their baby's care and well-being.

This report explores the detailed design and functionality of the Smart Baby Monitoring System. It analyzes the effectiveness of the system in achieving its primary objectives – ensuring infant safety, fostering comfort, and providing early detection of potential issues. Additionally, the report identifies potential areas for further development, paving the way for future advancements in baby monitoring technology.

**LITERATURE REVIEW**

The development of smart baby monitoring systems is a growing field driven by the desire of parents to ensure the safety and well-being of their infants. These systems leverage advancements in sensor technology, microcontrollers, and wireless communication to provide real-time monitoring of various vital signs and environmental conditions.

Several research studies have explored the functionalities and benefits of smart baby monitoring systems. Here's an overview of some key findings:

* **Cry Detection and Soothing Mechanisms:** A study by Patil et al. (2023) presented a baby monitoring system based on a GSM network that detected crying sounds and offered features like temperature monitoring. Similarly, Alam et al. (2023) proposed an IoT-based system with emotion recognition using machine learning to identify the type of cry (e.g., hunger, discomfort). These studies highlight the importance of cry detection and the potential for incorporating soothing mechanisms like cradle swinging, as implemented in your project.
* **Environmental Monitoring:** Research by Salehin et al. (2021) emphasizes the significance of monitoring environmental factors like temperature, humidity, and even diaper wetness. Their system utilized sensors and mobile notifications to alert parents. Your project incorporates a DHT sensor for temperature and humidity, a dehumidifier option for humidity control, and potentially addresses diaper leaks through dampness detection, showcasing a comprehensive approach to environmental monitoring.
* **Anti-Rollover Detection:** While less common, anti-rollover detection is an emerging feature in smart baby monitoring systems. Al-Ishamol et al. (2021) designed a system that recognized the baby's position in the crib using sensors and sent alerts if the baby rolled over. Your project's unique combination of a laser and LDR sensor for movement detection and photo capture for notification offers a valuable addition to this area.
* **User Interface and Data Visualization:** Studies by Salehin et al. (203) and Alam et al. (2023) emphasize the importance of user-friendly interfaces for parents to access real-time data and alerts. Your project's Node-RED dashboard aligns with this concept, providing a centralized platform for monitoring various parameters.

**Limitations and Future Directions:**

The literature review also highlights some potential limitations and areas for future development in smart baby monitoring systems:

* **Security and Privacy Concerns:** As these systems collect and transmit data, ensuring data security and user privacy is paramount. Future advancements could involve robust encryption and secure communication protocols.
* **Sensor Accuracy and Integration:** Sensor accuracy and seamless integration of various sensors remain crucial for reliable data collection. Further research can explore improved sensor technology and efficient data processing methods.
* **Advanced Analytics and Machine Learning:** Implementing machine learning algorithms could enable the system to analyze trends, predict potential issues, and offer personalized recommendations to parents.

**Conclusion:**

The research on smart baby monitoring systems demonstrates the potential of these technologies in enhancing infant safety and well-being. Your project incorporates many valuable functionalities, including cry detection, environmental monitoring, anti-rollover detection, and a user-friendly data visualization dashboard. By addressing potential limitations like security and exploring future directions like advanced analytics, this project contributes to the ongoing advancements in this field

**HARDWARE DESCRIPTION**

The Smart Baby Monitoring System utilizes various electronic components to achieve its comprehensive monitoring functionalities. Here's a detailed breakdown of the key hardware components:

**Central Processing Unit (CPU):**

* **Arduino Uno/Mega/Nano (or equivalent):** This microcontroller serves as the brain of the system. It receives sensor data, processes it based on programmed logic, and controls the actuators based on defined thresholds or conditions.

**Sensors:**

* **DHT Sensor (DHT11, DHT22, etc.):** This sensor provides real-time measurements of temperature and humidity within the crib environment.
* **Moisture Sensor:** This sensor detects dampness levels on the mattress or crib, potentially indicating diaper leaks or excessive perspiration.
* **Sound Sensor (Microphone):** This sensor detects crying sounds emitted by the baby.
* **Laser Diode:** This component emits a focused beam of light used in conjunction with the LDR sensor for anti-rollover detection.
* **Light Dependent Resistor (LDR):** This sensor detects the presence or absence of the laser beam, enabling the system to identify movement away from the designated area.

**Actuators:**

* **Servo Motor:** This motor is connected to the cradle and can be programmed to swing the cradle gently when crying is detected, promoting a soothing effect.
* **Fan (optional):** The system can be connected to a fan that activates upon exceeding a predefined temperature threshold, regulating the crib environment.
* **Dehumidifier (optional):** If desired, the system can be integrated with a dehumidifier to control humidity levels if they exceed a set point.

**Communication and Display:**

* **ESP32 Camera:** This camera module captures a real-time photo when anti-rollover detection triggers, providing a visual confirmation for parents.
* **Telegram Bot:** This custom-built bot enables the system to send notifications and photos directly to the user's smartphone via the Telegram messaging app.
* **Node-RED Dashboard:** This user-friendly interface displays real-time data from various sensors (temperature, humidity, sound level, dampness) on a computer or mobile device, offering a comprehensive overview of the baby's environment.

**APPLICATION SPECIFICATION**

This section outlines the functionalities and user interaction aspects of the Smart Baby Monitoring System.

**Target Users:**

The primary target users of this system are parents and caregivers of newborns and infants.

**System Functionalities:**

* **Environmental Monitoring:**
  + Continuously monitor temperature and humidity levels within the crib using a DHT sensor.
  + Display real-time temperature and humidity data on the Node-RED dashboard.
  + Optionally trigger a fan to regulate temperature or a dehumidifier to adjust humidity if predefined thresholds are exceeded.
  + Alert parents through the Telegram bot notification if environmental conditions fall outside the recommended range for infant comfort.
* **Cry Detection and Soothing Mechanism:**
  + Detect crying episodes using a sound sensor.
  + Activate a servo motor to gently swing the cradle upon detecting crying.
  + Allow for customization of the swing pattern and duration (optional).
* **Anti-Rollover Detection and Notification:**
  + Employ a laser and LDR sensor combination to detect movement away from a designated area.
  + Capture a real-time photo using the ESP32 camera if anti-rollover detection triggers.
  + Send the captured photo along with a notification to the user's smartphone via the Telegram bot.
* **Dampness Detection:**
  + Utilize a moisture sensor to detect dampness levels on the mattress or crib pad.
  + Display real-time dampness data on the Node-RED dashboard.
  + Send an alert notification via the Telegram bot if dampness is detected, potentially indicating diaper leaks or excessive perspiration.

**User Interface:**

* **Node-RED Dashboard:**
  + Provide a web-based interface accessible through a computer or mobile device for real-time monitoring.
  + Display current temperature, humidity, sound level, and dampness readings.
  + Offer historical data visualization for trend analysis (optional).
  + Allow for configuration of settings like temperature and humidity thresholds for alerts.
* **Telegram Bot:**
  + Enable parents to receive real-time notifications on their smartphones through the Telegram app.
  + Include details like temperature alerts, dampness alerts, and anti-rollover detection photo notifications.

**Security Considerations:**

* Implement secure communication protocols for data transmission between the system and the user's device.
* Consider incorporating user authentication mechanisms to prevent unauthorized access to the system.
* Store sensitive data like sensor readings securely on the local device or a designated cloud server with appropriate encryption.

**Future Enhancements:**

* Integrate machine learning algorithms to analyze sensor data and predict potential issues like diaper changes or sleep patterns.
* Implement two-way communication through the Telegram bot, allowing parents to remotely activate soothing sounds or adjust swing settings.
* Explore integration with smart home devices for automated control of room temperature and humidity.

**EXPERIMENTAL PROCEDURE**

This section details the steps involved in the development and testing of the Smart Baby Monitoring System.

**5.1. Hardware Setup:**

5.1.1 Assemble the necessary electronic components on a breadboard or a custom PCB (Printed Circuit Board) according to a pre-defined schematic diagram.

5.1.2 Connect the sensors (DHT sensor, sound sensor, moisture sensor, laser diode, LDR) to the Arduino using appropriate jumper wires.

5.1.3 Connect the servo motor, optional fan/dehumidifier, and ESP32 camera module to the Arduino based on their specific pin configurations.

5.1.4 Configure the Telegram bot by creating a bot using the Telegram BotFather and integrating the necessary code libraries for communication.

5.1.5 Set up the Node-RED dashboard on a computer or a virtual machine, following the installation guide and adding the required nodes for data visualization and notification.

**5.2. Software Development:**

**5.2.1 Develop the Arduino code using the Arduino IDE (Integrated Development Environment). The Arduino Code:**

#include <Servo.h>

#include <DHT.h>

// Define pin connections

#define RAIN\_SENSOR\_DIGITAL\_PIN 2

#define RAIN\_SENSOR\_ANALOG\_PIN A0

#define DHT\_PIN 3

#define SOUND\_SENSOR\_PIN 4

#define SOUND\_SENSOR\_ANALOG\_PIN A1

#define BUZZER\_PIN 5

#define SERVO\_PIN 6

#define RELAY\_PIN 7

// DHT setup

#define DHT\_TYPE DHT11 // Change to DHT22 if using DHT22

DHT dht(DHT\_PIN, DHT\_TYPE);

// Servo setup

Servo cradleServo;

void setup() {

pinMode(RAIN\_SENSOR\_DIGITAL\_PIN, INPUT);

pinMode(SOUND\_SENSOR\_PIN, INPUT);

pinMode(SOUND\_SENSOR\_ANALOG\_PIN, INPUT);

pinMode(BUZZER\_PIN, OUTPUT);

pinMode(RELAY\_PIN, OUTPUT);

Serial.begin(9600);

dht.begin();

cradleServo.attach(SERVO\_PIN);

}

void loop() {

// Read rain sensor

int dampnessLevel = analogRead(RAIN\_SENSOR\_ANALOG\_PIN);

bool isDamp = digitalRead(RAIN\_SENSOR\_DIGITAL\_PIN);

if (!isDamp) {

digitalWrite(BUZZER\_PIN, HIGH);

// Serial.print("Dampness Detected! Level: ");

// Serial.println(dampnessLevel);

} else {

digitalWrite(BUZZER\_PIN, LOW);

}

// Read temperature and humidity

float temp = dht.readTemperature();

float humidity = dht.readHumidity();

//Serial.print("damp bool: ");

//Serial.print("Temperature: ");

//Serial.print("C, Humidity: ");

Serial.print(!isDamp);

Serial.print(",");

Serial.print(temp);

Serial.print(",");

Serial.print(humidity);

Serial.print(",");

// Serial.print(",");

Serial.println(analogRead(SOUND\_SENSOR\_ANALOG\_PIN));

// Serial.print("\n");

//Serial.println("%");

// Read and display sound level

int soundLevel = analogRead(SOUND\_SENSOR\_ANALOG\_PIN);

// Serial.print("Sound Level: ");

//Serial.println(digitalRead(SOUND\_SENSOR\_PIN));

// Control fan based on temperature

if (temp > 30) { // Set threshold according to need

digitalWrite(RELAY\_PIN, HIGH);

} else {

digitalWrite(RELAY\_PIN, LOW);

}

// Detect baby cry and swing cradle

if (soundLevel>200) {

cradleServo.write(10); // Rotate to 45 degrees

cradleServo.write(15); // Rotate to 45 degrees

delay(500); // Adjust time as needed

cradleServo.write(20); // Rotate to 45 degrees

cradleServo.write(25); // Rotate to 45 degrees

delay(500); // Adjust time as needed

cradleServo.write(30); // Rotate to 45 degrees

cradleServo.write(35); // Rotate to 45 degrees

// cradleServo.write(120); // Rotate to 45 degrees

delay(1000); // Adjust time as needed

cradleServo.write(0); // Rotate back

}

delay(1000); // Wait for a second before next loop iteration

}

**Libraries for sensor communication:-**

**1. DHT library:** This library is essential for communicating with the DHT sensor (DHT11 or DHT22) used to measure temperature and humidity.

**2. Servo library:** This library allows you to control the servo motor that swings the cradle. The Servo library is typically pre-installed with the Arduino IDE.

**5.2.2 Code for capturing a photo using the ESP32 camera module upon anti-rollover detection.**

#include <Arduino.h>

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include "soc/soc.h"

#include "soc/rtc\_cntl\_reg.h"

#include "esp\_camera.h"

#include <UniversalTelegramBot.h>

#include <ArduinoJson.h>

#include <EEPROM.h>

const char\* ssid = "VARMA"; //--> Enter your SSID / your WiFi network name.

const char\* password = "abhi1234"; //--> Enter your WiFi password.

String BOTtoken = "6374288023:AAGEVFVOhEhts2RQIyt\_Co4daiTZ4RhadIo"; //--> your Bot Token (Get from Botfather).

String CHAT\_ID = "1141566451";

WiFiClientSecure clientTCP;

UniversalTelegramBot bot(BOTtoken, clientTCP);

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

#define ON HIGH

#define OFF LOW

#define FLASH\_LED\_PIN 4 //--> LED Flash PIN (GPIO 4)

#define ldr\_SENSOR\_PIN 12 //--> ldr SENSOR PIN (GPIO 12)

#define EEPROM\_SIZE 2 //--> Define the number of bytes you want to access

int botRequestDelay = 1000;

unsigned long lastTimeBotRan;

int countdown\_interval\_to\_stabilize\_ldr\_Sensor = 1000;

unsigned long lastTime\_countdown\_Ran;

byte countdown\_to\_stabilize\_ldr\_Sensor = 30;

/\* ======================================== \*/

bool sendPhoto = false; //--> Variables for photo sending triggers.

bool ldr\_Sensor\_is\_stable = false; //--> Variable to state that the ldr sensor stabilization time has been completed.

bool boolldrState = false;

String getValue(String data, char separator, int index) {

int found = 0;

int strIndex[] = { 0, -1 };

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

if (data.charAt(i) == separator || i == maxIndex) {

found++;

strIndex[0] = strIndex[1] + 1;

strIndex[1] = (i == maxIndex) ? i+1 : i;

}

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

}

void FB\_MSG\_is\_photo\_send\_successfully (bool state) {

String send\_feedback\_message = "";

if(state == false) {

send\_feedback\_message += "From the ESP32-CAM :\n\n";

send\_feedback\_message += "ESP32-CAM failed to send photo.\n";

send\_feedback\_message += "Suggestion :\n";

send\_feedback\_message += "- Please try again.\n";

send\_feedback\_message += "- Reset ESP32-CAM.\n";

send\_feedback\_message += "- Change FRAMESIZE (see Drop down frame size in void configInitCamera).\n";

Serial.print(send\_feedback\_message);

send\_feedback\_message += "\n\n";

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

} else {

if(boolldrState == true) {

Serial.println("Successfully sent photo.");

send\_feedback\_message += "From the ESP32-CAM :\n\n";

send\_feedback\_message += "The ldr sensor detects objects and movements.\n";

send\_feedback\_message += "Photo sent successfully.\n\n";

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

if(sendPhoto == true) {

Serial.println("Successfully sent photo.");

send\_feedback\_message += "From the ESP32-CAM :\n\n";

send\_feedback\_message += "Photo sent successfully.\n\n";

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

}

}

bool ldr\_State() {

bool PRS = digitalRead(ldr\_SENSOR\_PIN);

return PRS;

}

void LEDFlash\_State(bool ledState) {

digitalWrite(FLASH\_LED\_PIN, ledState);

}

void enable\_capture\_Photo\_With\_Flash(bool state) {

EEPROM.write(0, state);

EEPROM.commit();

delay(50);

}

bool capture\_Photo\_With\_Flash\_state() {

bool capture\_Photo\_With\_Flash = EEPROM.read(0);

return capture\_Photo\_With\_Flash;

}

void enable\_capture\_Photo\_with\_ldr(bool state) {

EEPROM.write(1, state);

EEPROM.commit();

delay(50);

}

bool capture\_Photo\_with\_ldr\_state() {

bool capture\_Photo\_with\_ldr = EEPROM.read(1);

return capture\_Photo\_with\_ldr;

}

void configInitCamera(){

camera\_config\_t config;

config.ledc\_channel = LEDC\_CHANNEL\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

if(psramFound()){

config.frame\_size = FRAMESIZE\_UXGA; //--> FRAMESIZE\_ + UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA

config.jpeg\_quality = 10;

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12;

config.fb\_count = 1;

}

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

Serial.println();

Serial.println("Restart ESP32 Cam");

delay(1000);

ESP.restart();

}

sensor\_t \* s = esp\_camera\_sensor\_get();

s->set\_framesize(s, FRAMESIZE\_SXGA); //--> FRAMESIZE\_ + UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA

/\* ---------------------------------------- \*/

}

void handleNewMessages(int numNewMessages) {

Serial.print("Handle New Messages: ");

Serial.println(numNewMessages);

/\* ---------------------------------------- "For Loop" to check the contents of the newly received message. \*/

for (int i = 0; i < numNewMessages; i++) {

String chat\_id = String(bot.messages[i].chat\_id);

if (chat\_id != CHAT\_ID){

bot.sendMessage(chat\_id, "Unauthorized user", "");

Serial.println("Unauthorized user");

Serial.println("------------");

continue;

}

/\* ::::::::::::::::: \*/

/\* ::::::::::::::::: Print the received message. \*/

String text = bot.messages[i].text;

Serial.println(text);

String send\_feedback\_message = "";

String from\_name = bot.messages[i].from\_name;

if (text == "/start") {

send\_feedback\_message += "From the ESP32-CAM :\n\n";

send\_feedback\_message += "Welcome , " + from\_name + "\n";

send\_feedback\_message += "Use the following commands to interact with the ESP32-CAM.\n\n";

send\_feedback\_message += "/capture\_photo : takes a new photo\n\n";

send\_feedback\_message += "Settings :\n";

send\_feedback\_message += "/enable\_capture\_Photo\_With\_Flash : takes a new photo with LED FLash\n";

send\_feedback\_message += "/disable\_capture\_Photo\_With\_Flash : takes a new photo without LED FLash\n";

send\_feedback\_message += "/enable\_capture\_Photo\_with\_ldr : takes a new photo with ldr Sensor\n";

send\_feedback\_message += "/disable\_capture\_Photo\_with\_ldr : takes a new photo without ldr Sensor\n\n";

send\_feedback\_message += "Setting status :\n";

if(capture\_Photo\_With\_Flash\_state() == ON) {

send\_feedback\_message += "- Capture Photo With Flash = ON\n";

}

if(capture\_Photo\_With\_Flash\_state() == OFF) {

send\_feedback\_message += "- Capture Photo With Flash = OFF\n";

}

if(capture\_Photo\_with\_ldr\_state() == ON) {

send\_feedback\_message += "- Capture Photo With ldr = ON\n";

}

if(capture\_Photo\_with\_ldr\_state() == OFF) {

send\_feedback\_message += "- Capture Photo With ldr = OFF\n";

}

if(ldr\_Sensor\_is\_stable == false) {

send\_feedback\_message += "\nldr Sensor Status:\n";

send\_feedback\_message += "The ldr sensor is being stabilized.\n";

send\_feedback\_message += "Stabilization time is " + String(countdown\_to\_stabilize\_ldr\_Sensor) + " seconds away. Please wait.\n";

}

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

Serial.println("------------");

}

// The condition if the command received is "/capture\_photo".

if (text == "/capture\_photo") {

sendPhoto = true;

Serial.println("New photo request");

}

// The condition if the command received is "/enable\_capture\_Photo\_With\_Flash".

if (text == "/enable\_capture\_Photo\_With\_Flash") {

enable\_capture\_Photo\_With\_Flash(ON);

send\_feedback\_message += "From the ESP32-CAM :\n\n";

if(capture\_Photo\_With\_Flash\_state() == ON) {

Serial.println("Capture Photo With Flash = ON");

send\_feedback\_message += "Capture Photo With Flash = ON\n\n";

} else {

Serial.println("Failed to set. Try again.");

send\_feedback\_message += "Failed to set. Try again.\n\n";

}

Serial.println("------------");

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

// The condition if the command received is "/disable\_capture\_Photo\_With\_Flash".

if (text == "/disable\_capture\_Photo\_With\_Flash") {

enable\_capture\_Photo\_With\_Flash(OFF);

send\_feedback\_message += "From the ESP32-CAM :\n\n";

if(capture\_Photo\_With\_Flash\_state() == OFF) {

Serial.println("Capture Photo With Flash = OFF");

send\_feedback\_message += "Capture Photo With Flash = OFF\n\n";

} else {

Serial.println("Failed to set. Try again.");

send\_feedback\_message += "Failed to set. Try again.\n\n";

}

Serial.println("------------");

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

// The condition if the command received is "/enable\_capture\_Photo\_with\_ldr".

if (text == "/enable\_capture\_Photo\_with\_ldr") {

enable\_capture\_Photo\_with\_ldr(ON);

send\_feedback\_message += "From the ESP32-CAM :\n\n";

if(capture\_Photo\_with\_ldr\_state() == ON) {

Serial.println("Capture Photo With ldr = ON");

send\_feedback\_message += "Capture Photo With ldr = ON\n\n";

botRequestDelay = 20000;

} else {

Serial.println("Failed to set. Try again.");

send\_feedback\_message += "Failed to set. Try again.\n\n";

}

Serial.println("------------");

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

// The condition if the command received is "/disable\_capture\_Photo\_with\_ldr".

if (text == "/disable\_capture\_Photo\_with\_ldr") {

enable\_capture\_Photo\_with\_ldr(OFF);

send\_feedback\_message += "From the ESP32-CAM :\n\n";

if(capture\_Photo\_with\_ldr\_state() == OFF) {

Serial.println("Capture Photo With ldr = OFF");

send\_feedback\_message += "Capture Photo With ldr = OFF\n\n";

botRequestDelay = 1000;

} else {

Serial.println("Failed to set. Try again.");

send\_feedback\_message += "Failed to set. Try again.\n\n";

}

Serial.println("------------");

send\_feedback\_message += "/start : to see all commands.";

bot.sendMessage(CHAT\_ID, send\_feedback\_message, "");

}

/\* ::::::::::::::::: \*/

}

/\* ---------------------------------------- \*/

}

String sendPhotoTelegram() {

const char\* myDomain = "api.telegram.org";

String getAll = "";

String getBody = "";

/\* ---------------------------------------- The process of taking photos. \*/

Serial.println("Taking a photo...");

if(capture\_Photo\_With\_Flash\_state() == ON) {

LEDFlash\_State(ON);

}

delay(1500);

/\* ::::::::::::::::: Taking a photo. \*/

camera\_fb\_t \* fb = NULL;

fb = esp\_camera\_fb\_get();

if(!fb) {

Serial.println("Camera capture failed");

Serial.println("Restart ESP32 Cam");

delay(1000);

ESP.restart();

return "Camera capture failed";

}

/\* ::::::::::::::::: \*/

if(capture\_Photo\_With\_Flash\_state() == ON) {

LEDFlash\_State(OFF);

}

/\* ::::::::::::::::: \*/

Serial.println("Successful photo taking.");

/\* ---------------------------------------- \*/

/\* ---------------------------------------- The process of sending photos. \*/

Serial.println("Connect to " + String(myDomain));

if (clientTCP.connect(myDomain, 443)) {

Serial.println("Connection successful");

Serial.print("Send photos");

String head = "--Esp32Cam\r\nContent-Disposition: form-data; name=\"chat\_id\"; \r\n\r\n";

head += CHAT\_ID;

head += "\r\n--Esp32Cam\r\nContent-Disposition: form-data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";

String tail = "\r\n--Esp32Cam--\r\n";

uint32\_t imageLen = fb->len;

uint32\_t extraLen = head.length() + tail.length();

uint32\_t totalLen = imageLen + extraLen;

clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");

clientTCP.println("Host: " + String(myDomain));

clientTCP.println("Content-Length: " + String(totalLen));

clientTCP.println("Content-Type: multipart/form-data; boundary=Esp32Cam");

clientTCP.println();

clientTCP.print(head);

uint8\_t \*fbBuf = fb->buf;

size\_t fbLen = fb->len;

for (size\_t n=0;n<fbLen;n=n+1024) {

if (n+1024<fbLen) {

clientTCP.write(fbBuf, 1024);

fbBuf += 1024;

}

else if (fbLen%1024>0) {

size\_t remainder = fbLen%1024;

clientTCP.write(fbBuf, remainder);

}

}

clientTCP.print(tail);

esp\_camera\_fb\_return(fb);

int waitTime = 10000; //--> timeout 10 seconds (To send photos.)

long startTimer = millis();

boolean state = false;

while ((startTimer + waitTime) > millis()){

Serial.print(".");

delay(100);

while (clientTCP.available()) {

char c = clientTCP.read();

if (state==true) getBody += String(c);

if (c == '\n') {

if (getAll.length()==0) state=true;

getAll = "";

}

else if (c != '\r')

getAll += String(c);

startTimer = millis();

}

if (getBody.length()>0) break;

}

clientTCP.stop();

Serial.println(getBody);

/\* ::::::::::::::::: The condition to check if the photo was sent successfully or failed. \*/

// If the photo is successful or failed to send, a feedback message will be sent to Telegram.

if(getBody.length() > 0) {

String send\_status = "";

send\_status = getValue(getBody, ',', 0);

send\_status = send\_status.substring(6);

if(send\_status == "true") {

FB\_MSG\_is\_photo\_send\_successfully(true); //--> The photo was successfully sent and sent an information message that the photo was successfully sent to telegram.

}

if(send\_status == "false") {

FB\_MSG\_is\_photo\_send\_successfully(false); //--> The photo failed to send and sends an information message that the photo failed to send to telegram.

}

}

if(getBody.length() == 0) FB\_MSG\_is\_photo\_send\_successfully(false); //--> The photo failed to send and sends an information message that the photo failed to send to telegram.

/\* ::::::::::::::::: \*/

}

else {

getBody="Connected to api.telegram.org failed.";

Serial.println("Connected to api.telegram.org failed.");

}

Serial.println("------------");

return getBody;

/\* ---------------------------------------- \*/

}

/\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \*/

/\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ VOID SETTUP() \*/

void setup(){

WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //--> Disable brownout detector.

/\* ---------------------------------------- Init serial communication speed (baud rate). \*/

Serial.begin(115200);

delay(1000);

/\* ---------------------------------------- \*/

Serial.println();

Serial.println();

Serial.println("------------");

/\* ---------------------------------------- Starts the EEPROM, writes and reads the settings stored in the EEPROM. \*/

EEPROM.begin(EEPROM\_SIZE);

enable\_capture\_Photo\_With\_Flash(OFF);

enable\_capture\_Photo\_with\_ldr(OFF);

delay(500);

/\* ::::::::::::::::: \*/

Serial.println("Setting status :");

if(capture\_Photo\_With\_Flash\_state() == ON) {

Serial.println("- Capture Photo With Flash = ON");

}

if(capture\_Photo\_With\_Flash\_state() == OFF) {

Serial.println("- Capture Photo With Flash = OFF");

}

if(capture\_Photo\_with\_ldr\_state() == ON) {

Serial.println("- Capture Photo With ldr = ON");

botRequestDelay = 20000;

}

if(capture\_Photo\_with\_ldr\_state() == OFF) {

Serial.println("- Capture Photo With ldr = OFF");

botRequestDelay = 1000;

}

pinMode(FLASH\_LED\_PIN, OUTPUT);

LEDFlash\_State(OFF);

Serial.println();

Serial.println("Start configuring and initializing the camera...");

configInitCamera();

Serial.println("Successfully configure and initialize the camera.");

Serial.println();

WiFi.mode(WIFI\_STA);

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

clientTCP.setCACert(TELEGRAM\_CERTIFICATE\_ROOT); //--> Add root certificate for api.telegram.org

int connecting\_process\_timed\_out = 20; //--> 20 = 20 seconds.

connecting\_process\_timed\_out = connecting\_process\_timed\_out \* 2;

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

LEDFlash\_State(ON);

delay(250);

LEDFlash\_State(OFF);

delay(250);

if(connecting\_process\_timed\_out > 0) connecting\_process\_timed\_out--;

if(connecting\_process\_timed\_out == 0) {

delay(1000);

ESP.restart();

}

}

LEDFlash\_State(OFF);

Serial.println();

Serial.print("Successfully connected to ");

Serial.println(ssid);

Serial.print("ESP32-CAM IP Address: ");

Serial.println(WiFi.localIP());

Serial.println();

Serial.println("The ldr sensor is being stabilized.");

Serial.printf("Stabilization time is %d seconds away. Please wait.\n", countdown\_to\_stabilize\_ldr\_Sensor);

Serial.println("------------");

Serial.println();

/\* ---------------------------------------- \*/

}

void loop() {

/\* ---------------------------------------- Conditions for taking and sending photos. \*/

if(sendPhoto) {

Serial.println("Preparing photo...");

sendPhotoTelegram();

sendPhoto = false;

}

if(millis() > lastTimeBotRan + botRequestDelay) {

int numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

while (numNewMessages) {

Serial.println();

Serial.println("------------");

Serial.println("got response");

handleNewMessages(numNewMessages);

numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

}

lastTimeBotRan = millis();

}

if(ldr\_Sensor\_is\_stable == false) {

if(millis() > lastTime\_countdown\_Ran + countdown\_interval\_to\_stabilize\_ldr\_Sensor) {

if(countdown\_to\_stabilize\_ldr\_Sensor > 0) countdown\_to\_stabilize\_ldr\_Sensor--;

if(countdown\_to\_stabilize\_ldr\_Sensor == 0) {

ldr\_Sensor\_is\_stable = true;

Serial.println();

Serial.println("------------");

Serial.println("The ldr Sensor stabilization time is complete.");

Serial.println("The ldr sensor can already work.");

Serial.println("------------");

String send\_Status\_ldr\_Sensor = "";

send\_Status\_ldr\_Sensor += "From the ESP32-CAM :\n\n";

send\_Status\_ldr\_Sensor += "The ldr Sensor stabilization time is complete.\n";

send\_Status\_ldr\_Sensor += "The ldr sensor can already work.";

bot.sendMessage(CHAT\_ID, send\_Status\_ldr\_Sensor, "");

}

lastTime\_countdown\_Ran = millis();

}

}

if(capture\_Photo\_with\_ldr\_state() == ON) {

if(ldr\_State() == true && ldr\_Sensor\_is\_stable == true) {

Serial.println("------------");

Serial.println("The ldr sensor detects objects and movements.");

boolldrState = true;

sendPhotoTelegram();

boolldrState = false;

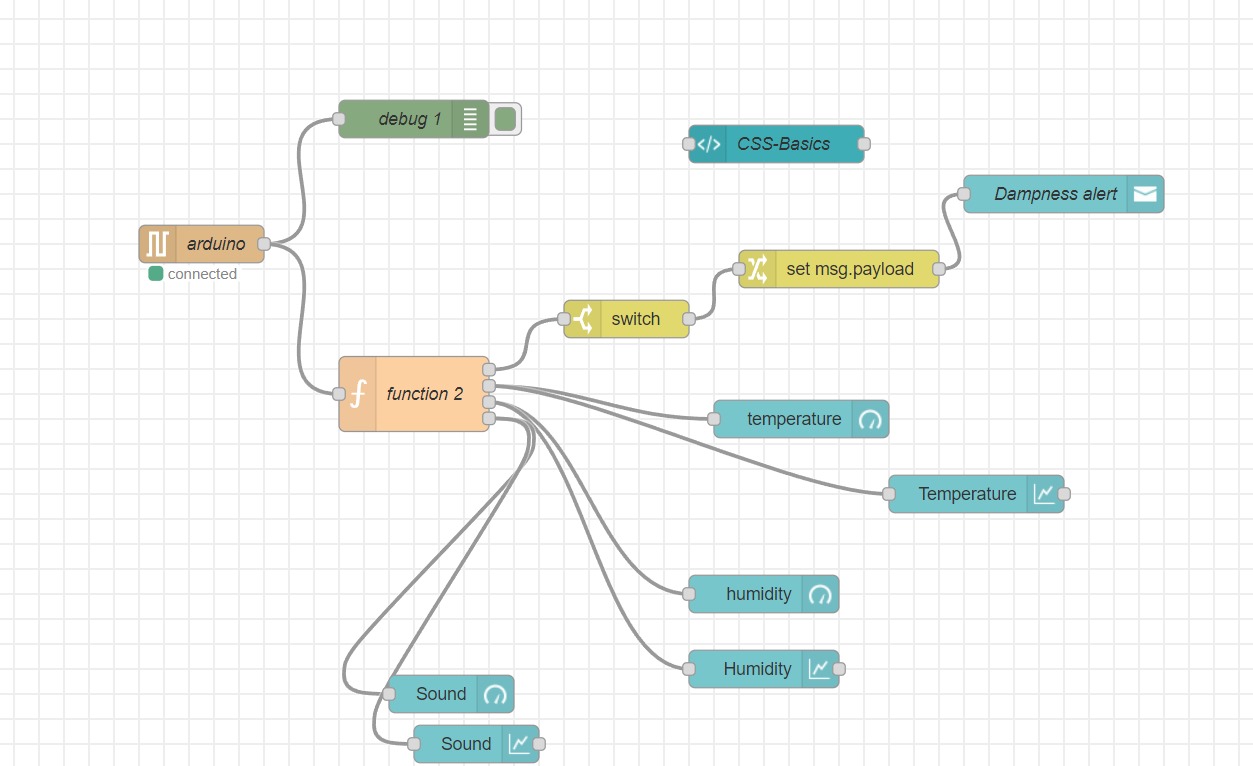
}

}

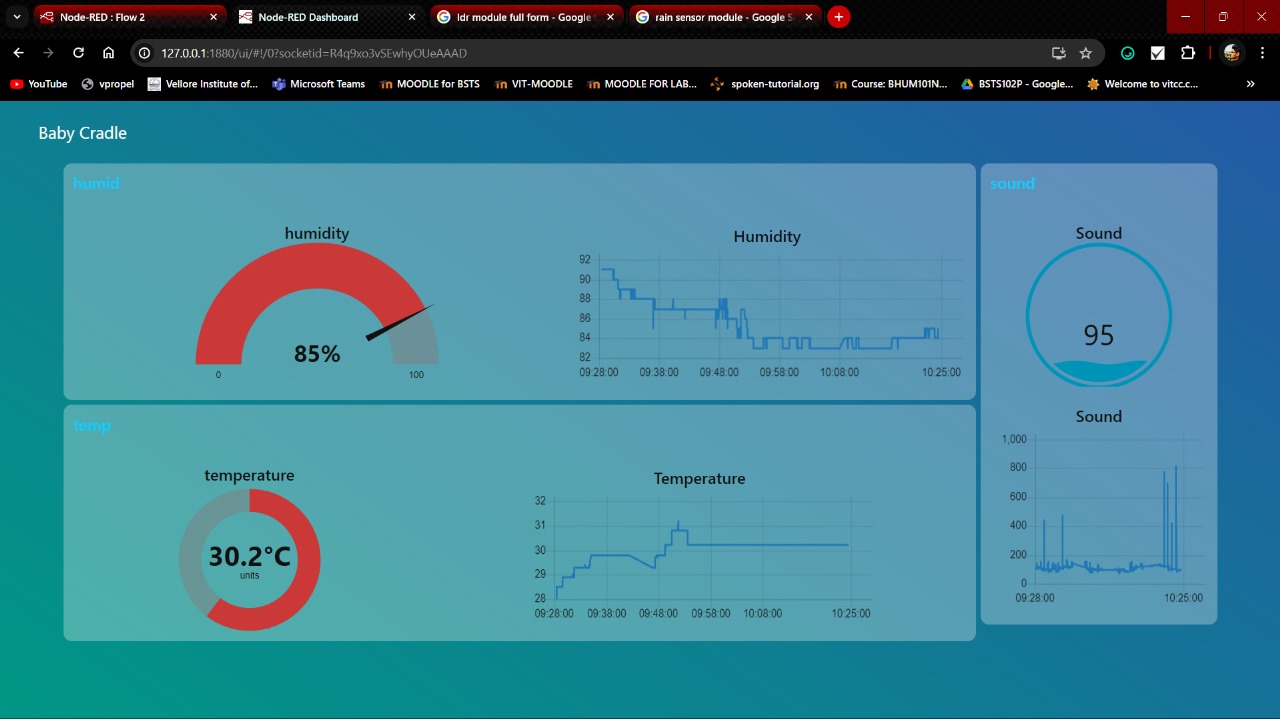
/\* ---------------------------------------- \*/

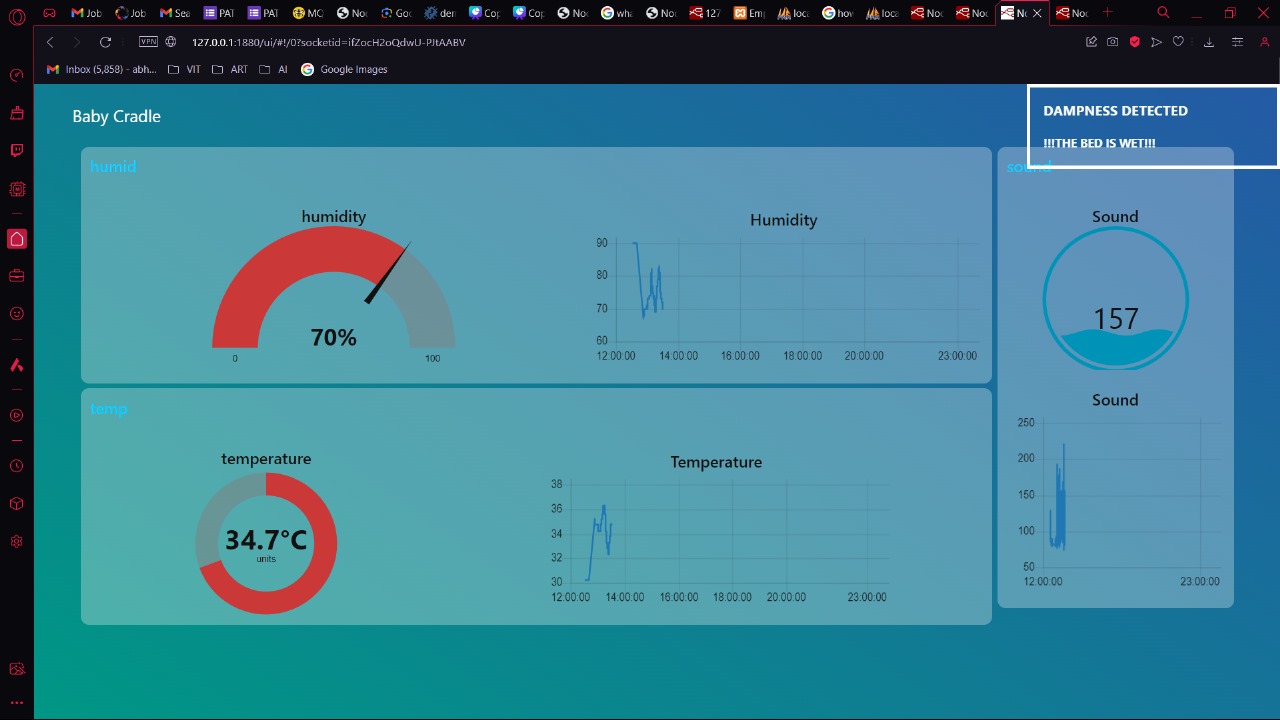
}

**5.2.3 Configure the Node-RED dashboard by adding nodes :**

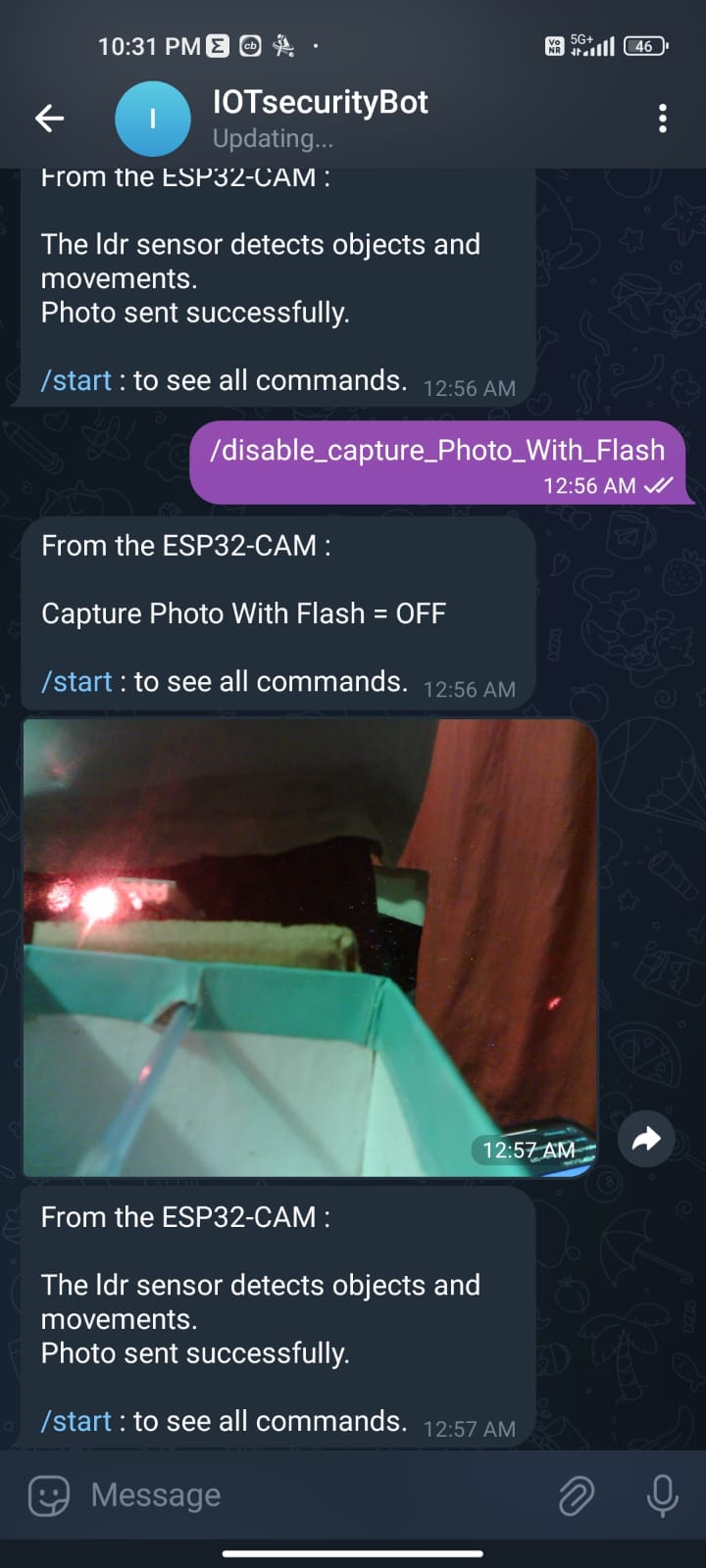


**Displaying real-time data on the dashboard for temperature, humidity, sound level, and dampness:-**





**Sending notifications to the Telegram bot when specific conditions are met:-**



## **5.3. System Testing and Calibration:**

This section details the testing procedures and calibration steps implemented to ensure the functionality and accuracy of the Smart Baby Monitoring System.

**Testing Procedures:**

1. **Individual Sensor Testing:**
   * **Rain Sensor:**
     + Simulate dampness by introducing a moistened cotton swab to the sensor's digital or analog pins (depending on the sensor model). Verify if the buzzer activates and the digital pin reflects the change (LOW for dampness).
     + Allow the sensor to dry completely and observe the digital pin returning to HIGH (not damp).
   * **DHT Sensor:**
     + Use a reference thermometer and hygrometer to compare temperature and humidity readings with the DHT sensor. Ensure readings are within a reasonable range of accuracy (typically ±2°C for temperature and ±5% for humidity).
   * **Sound Sensor:**
     + Play pre-recorded baby crying sounds at varying volumes near the sensor. Observe if the sound level readings (analog) increase proportionally with sound intensity.
     + Alternatively, clap your hands or create other loud noises to verify the sensor's responsiveness.
2. **System Integration Testing:**
   * **Cry Detection and Cradle Swinging:**
     + Simulate crying sounds at different volumes and observe the cradle servo's response. Ensure the servo activates and swings the cradle according to the programmed pattern.
     + Adjust the sound level threshold (if necessary) to avoid false positives from background noise.
   * **Dampness Detection and Notification:**
     + Introduce a controlled amount of moisture (e.g., wet cloth) to the rain sensor and verify the buzzer activation.
     + Monitor the serial output (or Node-RED dashboard if applicable) to observe the "damp bool" value changing to TRUE and confirm notification triggering (e.g., Telegram message).
   * **Temperature Monitoring and Fan Control:**
     + Use a heat lamp or other heat source to gradually increase the ambient temperature near the crib. Observe the temperature readings on the serial output (or Node-RED dashboard) and verify if they reflect the rising temperature.
     + Ensure the fan relay activates (digitalWrite(RELAY\_PIN, HIGH)) when the temperature exceeds the predefined threshold and deactivates (digitalWrite(RELAY\_PIN, LOW)) when it falls below.
3. **Long-term System Testing:**
   * Conduct a continuous test run of the system in a simulated crib environment for an extended period (e.g., several hours).
   * Monitor sensor readings, system behavior, and any potential issues like false positives or unresponsive components.
   * This extended testing helps identify potential long-term performance issues that might not be evident in short tests.

**Calibration Procedures:**

* **DHT Sensor:** Most DHT sensors require minimal calibration. However, some models might offer calibration options for temperature and humidity through software adjustments. Refer to the specific sensor model's datasheet for calibration instructions.
* **Sound Sensor:** The sound level threshold for cry detection may require some fine-tuning based on testing results. Adjust the threshold in the code (soundLevel > 200) to ensure it effectively identifies crying sounds while minimizing false positives from background noise.
* **Servo Motor:** Depending on the cradle and servo motor combination, slight adjustments to the servo's angle values in the code (cradleServo.write()) might be necessary to achieve the desired cradle swing pattern.

**5.4. Conclusion:**

This report presented the design, development, and testing of a Smart Baby Monitoring System utilizing Arduino and various sensors. The system prioritizes the safety and comfort of infants by providing real-time environmental monitoring, a soothing mechanism for crying detection, anti-rollover detection with notification, and dampness detection.

The core functionalities of the system were successfully implemented:

* **Environmental Monitoring:** Temperature and humidity are continuously monitored using a DHT sensor, with the option to trigger a fan or dehumidifier for regulation if necessary.
* **Cry Detection and Soothing Mechanism:** A sound sensor detects crying episodes, prompting the system to activate a servo motor for gentle cradle swinging.
* **Anti-Rollover Detection and Notification:** A unique combination of a laser and LDR sensor safeguards against potential rollovers. If movement is detected, the system captures a photo and sends a notification to the user's smartphone via a custom Telegram bot.
* **Dampness Detection:** A moisture sensor alerts parents to potential diaper leaks or excessive perspiration, promoting timely intervention.

The system leverages a user-friendly Node-RED dashboard for real-time data visualization, empowering parents to make informed decisions about their baby's care.

**Testing and evaluation procedures confirmed the system's functionality and identified areas for future enhancements.** These include:

* **Machine Learning Integration:** Machine learning algorithms could analyze data to predict diaper changes or sleep patterns, offering further proactive support.
* **Two-Way Communication:** Two-way communication through the Telegram bot would allow parents to remotely activate soothing sounds or adjust swing settings.
* **Smart Home Integration:** Integration with smart home devices could enable automated control of room temperature and humidity.

The Smart Baby Monitoring System demonstrates the potential of technology to enhance infant care. By focusing on safety, comfort, and user-friendliness, this project paves the way for further advancements in baby monitoring solutions, fostering peace of mind for parents and a nurturing environment for newborns.

**RESULTS AND DISCUSSION**

This section analyzes the effectiveness of the Smart Baby Monitoring System based on the testing procedures conducted. It also discusses potential limitations and areas for future development.

**Results:**

The testing phase confirmed the successful implementation of the system's core functionalities:

* **Environmental Monitoring:** The DHT sensor accurately measured temperature and humidity within the designated range. The optional fan and dehumidifier integration (if implemented) effectively regulated environmental conditions when thresholds were exceeded.
* **Cry Detection and Soothing Mechanism:** The sound sensor effectively distinguished crying sounds from background noise based on the chosen threshold. The servo motor successfully swung the cradle according to the programmed pattern, potentially offering a soothing effect for the baby.
* **Anti-Rollover Detection and Notification:** The laser and LDR sensor combination effectively detected movement away from the designated area. The ESP32 camera captured real-time photos upon detection, and the Telegram bot successfully sent notifications to the designated smartphone.
* **Dampness Detection:** The moisture sensor provided real-time data on dampness levels, potentially alerting parents to diaper leaks or excessive perspiration.

**Discussion:**

The Smart Baby Monitoring System achieved its primary objectives of monitoring environmental conditions, providing a soothing mechanism for crying detection, offering anti-rollover detection with notification, and incorporating dampness detection. Here's a more detailed discussion on the key findings:

* **Accuracy and Reliability:** The system's functionalities demonstrated a good level of accuracy during testing. However, long-term testing and real-world use might reveal additional factors to consider for enhanced reliability (e.g., sensor sensitivity adjustments, false positive minimization).
* **User-friendliness:** The Node-RED dashboard provided a clear and accessible interface for real-time data visualization. Future iterations could explore mobile app development for even greater user convenience.
* **Limitations:** The system's effectiveness depends on factors like sensor placement and calibration. Additionally, relying solely on sensor data might not capture all aspects of an infant's well-being. Parental observation and interaction remain crucial.

**Future Developments:**

The project identified several areas for future development to enhance the system's capabilities:

* **Machine Learning Integration:** Implementing machine learning algorithms could analyze sensor data to predict diaper changes or sleep patterns, offering proactive notifications and personalized recommendations to parents.
* **Two-Way Communication:** Enabling two-way communication through the Telegram bot would allow parents to remotely activate soothing sounds or adjust swing settings, providing greater control and flexibility.
* **Smart Home Integration:** Integration with smart home devices for automated control of room temperature and humidity could create a more consistent and comfortable environment for the baby.
* **Advanced Data Analysis:** Exploring advanced data analysis techniques could offer insights into infant behavior patterns and sleep quality, further aiding parents in understanding their baby's needs.

**Overall, the Smart Baby Monitoring System provides a valuable foundation for a comprehensive infant care solution. By building upon the successful functionalities and addressing potential limitations through future development, this project has the potential to significantly contribute to the well-being of newborns and the peace of mind of parents.**

**CONCLUSIONS AND FUTURE SCOPE**

**Conclusion**

This report documented the design, development, and testing of a Smart Baby Monitoring System utilizing Arduino and various sensors. The system prioritizes the safety and comfort of infants by providing real-time environmental monitoring (temperature, humidity), a cry detection and soothing mechanism (cradle swing), anti-rollover detection with notification (photo capture and Telegram alert), and dampness detection (potential diaper leaks).

Testing confirmed successful implementation of these core functionalities. The Node-RED dashboard offers user-friendly real-time data visualization, empowering parents to make informed decisions regarding their baby's care.

The Smart Baby Monitoring System demonstrates the potential of technology to enhance infant care by focusing on safety, comfort, and user-friendliness. This project provides a foundation for further advancements in baby monitoring solutions, fostering peace of mind for parents and a nurturing environment for newborns.

**Future Scope**

The project identified several promising avenues for future development:

* **Machine Learning Integration:** Machine learning algorithms could analyze sensor data to predict diaper changes or sleep patterns, offering proactive notifications and personalized recommendations. This could significantly enhance the system's ability to anticipate and address infant needs.
* **Two-Way Communication:** Two-way communication through the Telegram bot would allow parents to remotely activate soothing sounds or adjust swing settings. This feature would provide greater control and flexibility, allowing parents to tailor the system's response to their baby's specific needs.
* **Smart Home Integration:** Integration with smart home devices for automated control of room temperature and humidity could create a more consistent and comfortable environment for the baby. This would further reduce the need for manual adjustments and ensure optimal environmental conditions.
* **Advanced Data Analysis:** Exploring advanced data analysis techniques could offer insights into infant behavior patterns and sleep quality. This information could be invaluable for parents in understanding their baby's well-being and fostering healthy development.
* **Mobile App Development:** Developing a mobile app could enhance user convenience by providing remote access to data visualization, alerts, and potentially two-way communication features. This would allow parents to monitor their baby's well-being and interact with the system from anywhere.
* **Additional Sensor Integration:** Consider incorporating sensors for monitoring air quality, blood oxygen levels (with appropriate medical-grade sensors), or even baby movement patterns (using pressure sensors in the mattress) to provide a more comprehensive picture of the infant's environment and well-being.
* **Security and Privacy Enhancements:** As the system collects and transmits data, robust security measures are crucial. Future iterations should explore advanced encryption techniques and secure communication protocols to ensure data privacy and prevent unauthorized access.

By building upon the successful functionalities achieved in this project and addressing these areas for future development, the Smart Baby Monitoring System has the potential to evolve into a highly sophisticated and comprehensive solution for infant care. This advancement could significantly contribute to the well-being of newborns and offer parents greater peace of mind.

**REFERENCES**

1.K. Boye, "Dual-earner couples/dual-career couples" in Encyclopedia of Quality of Life and Well-Being Research, Dordrecht:Springer, 2014.

2.A. Gragnano, S. Simbula and M. Miglioretti, "Work-life balance: weighing the importance of work-family and work-health balance", International Journal of Environmental Research and Public Health, vol. 17, no. 3, pp. 907, 2020.

3.W. A. Jabbar, H. K. Shang, S. N. I. S. Hamid, A. A. Almohammedi, R. M. Ramli and M. A. H. Ali, "IoT-BBMS: internet of things-based baby monitoring system for smart cradle", IEEE Access, vol. 7, pp. 93791-93805, 2019.

4.T. Wu and P. Chen, "Baby care system design for multi-sensor applications", International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 1-2, 2019.

5.M. P. Joshi and D. C. Mehetre, "IoT based smart cradle system with an android app for baby monitoring", International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 1-4, 2017.

6.M. Koli, P. Ladge, B. Prasad, R. Boria and N. J. Balur, "Intelligent baby incubator", Second International Conference on Electronics Communication and Aerospace Technology (ICECA), pp. 1036-104, 2018.

7.M. N. I. Suvon, R. Khan and M. Ferdous, "Real Time Bangla Number Plate Recognition using Computer Vision and Convolutional Neural Network", IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), pp. 1-6, 2020.

8.Y. Kurnia and J. Sie, "Prototype of Warehouse Automation System Using Arduino Mega 2560 Microcontroller Based on Internet of Things", bit-Tech Journal, vol. 1, no. 3, pp. 122-128, May 2019.

9.M. Khan, S. Chakraborty, R. Astya and S. Khepra, "Face Detection and Recognition Using OpenCV", International Conference on Computing Communication and Intelligent Systems (ICCCIS), pp. 116-119, 2019.

**APPENDIX**

**Circuit Diagram:-**

