ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

An Autonomous Robotic System for Multi-Modal, Non-Invasive Sensing and Early Illness Detection in Seven-Month-Old Infants

Moses Maduka Testimony[1]; Litsin Konstantin Vladimirovich[2]

[1]Department: Mechatronics and Robotics Universty: South Ural State University. (National Research University Russia)
[2]University: South Ural State University (National Research University)
[2]Associate Professor (PhD), Candidate of Technical Science, Department of Electric Drive, Mechatronics and Electromechanics.

Email: madumose1470@gmail.com.

Publication Date: 2025/10/31

Abstract: The early detection of illness in non-verbal infants, particularly at seven months of age, presents a significant challenge in pediatric care. Pre-verbal infants cannot articulate discomfort, leading to potential delays in diagnosis and treatment. This paper proposes the design and methodology for a novel, low-cost, and non-invasive monitoring system that leverages the Arduino microcontroller platform integrated with a thermal camera and passive infrared (PIR) sensors to create an early warning system for infant sickness. The system operates by continuously and unobtrusively monitoring two key physiological and behavioral correlates of illness: elevated core body temperature (fever) and alterations in sleep/wake activity patterns. The thermal camera (MLX90640) is employed to map facial temperature, identifying febrile states without physical contact. Concurrently, PIR sensors track gross motor activity and restlessness, which are often suppressed or increased during illness. Data from these sensors are processed by an Arduino Mega, which uses a rule-based algorithm to flag anomalies. If a potential sickness state is detected (e.g., sustained elevated temperature coupled with abnormal inactivity), the system triggers an alert to caregivers via a connected mobile application. This multi-modal approach aims to reduce false positives compared to single-parameter systems and provides a crucial tool for proactive parental intervention, potentially improving health outcomes for vulnerable infants.

Keywords: *Neonatal Monitoring, Non-Invasive Diagnostics, Arduino, Thermal Imaging, Infrared Sensor, Affective Computing, Early Warning System.*

How to Cite: Moses Maduka Testimony; Litsin Konstantin Vladimirovich (2025) An Autonomous Robotic System for Multi-Modal, Non-Invasive Sensing and Early Illness Detection in Seven-Month-Old Infants.
International Journal of Innovative Science and Research Technology,
10(10), 1951-1960. https://doi.org/10.38124/ijisrt/25oct519

I. INTRODUCTION

The advent of parenthood is accompanied by an innate, vigilant concern for an infant's well-being, a concern that is most acute during the pre-verbal stage of development. A seven-month-old infant exists in a critical transitional period: they are increasingly mobile, demonstrating new abilities like rolling and crawling, and are becoming more socially interactive, yet they remain entirely incapable of verbally communicating their physical state. This profound communication barrier creates a significant vulnerability, as common illnesses such as otitis media, respiratory syncytial virus (RSV), and urinary tract infections often manifest through non-specific signs like fever, lethargy, or irritability. These subtle, early indicators can be easily overlooked amidst the normal fluctuations of infant behavior or mistaken for teething or minor fussiness. Consequently, parents and caregivers may not recognize the need for medical consultation until symptoms have progressed to more severe states, such as persistent high fever, dehydration, or pronounced lethargy, potentially leading to complications and more intensive treatments.

Current standard practice for at-home health monitoring remains largely reactive and intermittent. It relies heavily on tools like tympanic or temporal thermometers, which require physical contact, can disturb a sleeping infant, and provide

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

only a single, isolated data point in time. This snapshot approach is fundamentally ill-suited for capturing the dynamic onset or progression of a febrile illness or the gradual decline in activity that signals malaise. The limitations of this paradigm highlight a clear and urgent need for a continuous, passive, and multi-parameter monitoring solution that can operate seamlessly within the infant's environment, particularly during sleep.

The convergence of accessible open-source hardware, advanced low-cost sensor technology, and the principles of affective computing offers a promising pathway to bridge this gap. Platforms like Arduino provide a robust and flexible foundation for integrating diverse sensors, while thermal imaging cameras, once confined to industrial and clinical settings, have been validated for non-contact fever screening, providing an accurate and hygienic method for temperature measurement. Simultaneously, the correlation between physical activity levels and health status is a well-documented behavioral marker, with lethargy and restlessness serving as reliable proxies for underlying sickness in infants

This article details the design and operational process of an autonomous, integrated system that synthesizes these technological advancements into a cohesive whole. By fusing thermal data for non-contact core temperature estimation with infrared-based activity monitoring, the proposed system moves beyond single-point measurements to provide a holistic, context-aware, and continuous assessment of an infant's well-being. The primary objective is to empower parents with real-time, data-driven insights that complement their caregiving intuition, transforming infant healthcare monitoring from a reactive chore into a proactive, intelligent safeguard.

II. SYSTEM DESIGN AND OPERATIONAL PROCESS

The proposed system is built around an Arduino Mega 2560 microcontroller, chosen for its multiple serial ports and sufficient I/O pins to handle concurrent sensor data streams. The system is designed to be mounted in the infant's sleeping area (e.g., above the crib) for continuous monitoring.

A. Hardware Components and their Roles

- *Microcontroller: Arduino Mega 2560.*

- Role: The central brain of the system. It reads data from all sensors, processes it using a predefined algorithm, and manages the output/alert system.

- *Thermal Imaging Camera: MLX90640 ESF-BAA 32x24 IR Array.*

- Role: To capture a low-resolution thermal map of the infant's face and body. The camera detects infrared radiation, which is converted into temperature values for each of the 768 pixels (32x24). By identifying the highest temperature region (which typically corresponds to the tear duct/inner canthus of the eye, a reliable proxy for core body temperature [4]), the system can estimate if the infant has a fever.

- *Passive Infrared (PIR) Sensor: HC-SR501.*

- Role: To detect motion within its field of view. One or more sensors are positioned to cover the crib area. During designated sleep times, a lack of expected motion (suggesting lethargy) or an excess of motion (suggesting restlessness/ discomfort) can be flagged as anomalous behavior.

- *Ambient Temperature & Humidity Sensor: DHT22.*

- Role: To provide environmental context. This is crucial for calibrating the thermal camera readings, as ambient temperature can affect perceived skin temperature.

- *Alert Module: ESP-01 ESP8266 WiFi Module.*

- Role: To connect the Arduino to the local WiFi network. When the algorithm detects a potential sickness event, the Arduino sends a command to the ESP8266 to transmit an alert to a designated smartphone application or web dashboard.

- *Power Supply: A Stable 5V DC power adapter to ensure continuous operation.*

B. Software and Data Processing Workflow

The system's intelligence lies in its software algorithm, which follows a multi-stage process:

- *Data Acquisition:*

- The MLX90640 captures a thermal frame every 10 seconds.
- The PIR sensor's digital output is read continuously to log motion events.
- The DHT22 provides an ambient temperature reading every 30 seconds.

- *Data Pre-Processing & Feature Extraction:*

- Thermal Data: The Arduino scans the 32x24 thermal grid to find the cluster of pixels with the highest average temperature, corresponding to the infant's facial region (specifically the inner canthus). This value is stored as T_core_estimate.
- Motion Data: The PIR data is analyzed over a 15-minute rolling window. The system calculates a "Motion Index" (e.g., seconds of activity per minute).

- *Anomaly Detection Algorithm (Rule-Based Logic):*

The microcontroller runs a continuous loop comparing the extracted features against predefined, customizable thresholds.

- Fever Check: If (T_core_estimate > 38.0°C) for 3 consecutive readings, Fever_Flag = TRUE.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

- *Then Activity Anomaly Check (During Sleep Periods):*

✓ If (Motion_Index < 0.1) for 30 minutes, then Lethargy_Flag = TRUE. (Extreme inactivity)
✓ If (Motion_Index > 2.0) for 30 minutes, then Restlessness_Flag = TRUE. (Excessive movement)

➢ *Alert Triggering:*
The final decision logic is a weighted combination:

- ALERT LEVEL 1 (Monitor Closely): Fever_Flag = TRUE OR Lethargy_Flag = TRUE.
- ALERT LEVEL 2 (Urgent): (Fever_Flag = TRUE) AND (Lethargy_Flag = TRUE).

Upon triggering an alert, the Arduino sends a JSON packet via the ESP8266 to a cloud service (e.g., Blynk, Adafruit IO), which pushes a notification to the parent's smartphone: *"Alert: Potential sickness detected. Elevated temperature and low activity."*

C. Implementation and Safety Considerations

- Calibration: The thermal camera must be calibrated against a clinical-grade thermometer in a controlled environment.
- Positioning: Sensors must be positioned to maximize coverage of the crib while minimizing false triggers from outside the monitored zone.
- Data Privacy: All data transmitted via WiFi must be encrypted. Local data storage on an SD card can be added as a backup.
- Safety: The entire system must be powered via a safe, wall-mounted adapter and placed securely out of the infant's reach. It is a monitoring aid, not a medical device, and is intended to supplement, not replace, parental vigilance and professional medical advice.

➢ *Detailed Process and Sensor Breakdown for Infant Health Monitoring System*
This section expands on the system's operation, breaking it down into a continuous cycle of data acquisition, processing, and decision-making.

➢ *The Continuous Monitoring Cycle*
The system operates on a 24/7 loop, with a primary focus on sleep periods when the infant is stationary and baseline measurements are most consistent. The entire process can be visualized in five key stages:

➢ *Sensing →Pre-Processing →Feature Extraction → Decision Logic →Alert & Action*

- Let's Detail Each Stage.

➢ *Stage 1: Sensing - Data Acquisition from the Environment*
This is the hardware layer where physical phenomena (heat, motion) are converted into electrical signals.

- *List of Sensors and their Detailed Functions:*

Table 1 Sensing - Data Acquisition from the Environment

Sensor Name & Model	Primary Function	Specific Use in This Project	Technical Details
Thermal Camera *(MLX90640 ESF-BAA)*	To capture a 2D map of temperature distribution without contact.	To non-invasively estimate the infant's core body temperature by measuring the skin temperature at the inner canthus (tear duct) of the eye, which is the most reliable external proxy for core temperature.	Type: Far Infrared (FIR) sensor array. Resolution: 32 pixels x 24 pixels (768 individual temperature points). Field of View: 110° x 75° (wide angle to cover the crib). Output: A array of 768 raw values representing object temperature.
Passive Infrared (PIR) Motion Sensor *(HC-SR501)*	To detect movement of infrared-radiating bodies (like humans).	To monitor the infant's gross motor activity and sleep patterns. It distinguishes between states of sleep (low motion), active sleep (some motion), and wakefulness/restlessness (high motion).	Type: Passive Infrared. Principle: Detects changes in infrared radiation in its field of view. It does not measure amount of motion, only its occurrence. Output: A digital signal (HIGH/LOW) indicating motion detection.
Ambient Temperature & Humidity Sensor *(DHT22 / AM2302)*	To measure the surrounding air temperature and relative humidity.	To provide environmental context for the thermal camera. Skin temperature readings can be influenced by room temperature. This data is used to improve the accuracy of the fever detection algorithm.	Type: Digital Sensor with a capacitive humidity sensor and a thermistor. Range: Temperature: -40 to 80°C; Humidity: 0-100% RH. Output: Digital signal with temperature and humidity values.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

WiFi Module *(ESP-01 ESP8266)*	To provide network connectivity to the Arduino.	To transmit system status and critical alerts from the Arduino to a cloud server, which then pushes notifications to a parent's smartphone app.	Type: SoC with integrated TCP/IP protocol stack. Interface: UART (Serial) with Arduino. Function: Acts as a wireless communication bridge.

➢ *Stage 2: Pre-Processing - Cleaning and Preparing the Data*

Raw sensor data is often noisy and needs to be filtered and calibrated before it can be used.

- *Thermal Data Pre-Processing:*

✓ Read Raw Data: The Arduino reads the 768 values from the MLX90640 over the I2C communication protocol.
✓ Ambient Compensation: The raw sensor readings are adjusted using the ambient temperature value from the DHT22. The MLX90640's own library often handles this using the sensor's own internal ambient temperature measurement, but the DHT22 provides a reliable secondary check.
✓ Noise Filtering (Software): A simple moving average filter is applied to reduce "spiky" noise in the temperature readings. For example, each pixel's temperature is averaged with its previous two readings to smooth the data.

- *Motion Data Pre-Processing:*

✓ Debouncing: The PIR sensor's digital signal is "debounced" in software to avoid counting a single motion event multiple times due to electrical noise.
✓ Quantification: The Arduino doesn't just see "motion" or "no motion." It counts the number of times the PIR sensor is triggered (HIGH) within a specific time window (e.g., 15 minutes). This creates a "Motion Count" metric.

➢ *Stage 3: Feature Extraction - Converting Data into Meaningful Metrics*

In this stage, the cleaned data is analyzed to extract specific, useful metrics that the algorithm can understand.

- *Detailed Thermal Imaging Process:*

This is the most complex part of the data processing.

✓ *Target Identification (Region of Interest - ROI):*

The system scans the entire 32x24 thermal grid.

It looks for a cluster of pixels (e.g., a 3x3 block) that is significantly warmer than the surrounding areas and is located in the expected upper-middle region of the frame (where the infant's face should be).

This warm spot is identified as the inner canthus of the eye, which is consistently the warmest point on the face and has a high correlation with core body temperature.

✓ *Temperature Calculation:*

The temperatures of the pixels within this identified ROI are averaged. This average is stored as T_core_estimate.

If the 3x3 eye-region pixels have temperatures [37.2, 37.5, 37.4, 37.6, 37.8, 37.5, 37.3, 37.4, 37.6]°C, the T_core_estimate would be (sum / 9) = 37.48°C.

✓ *Motion Index Calculation:*

The "Motion Count" from the PIR sensor over the last 15 minutes is converted into a "Motion Index" (Motion_Index = Total_Motion_Triggers / 15). This gives a normalized value for activity per minute.

45 triggers in 15 minutes = Motion Index of 3.0 (very active). 3 triggers in 15 minutes = Motion Index of 0.2 (very still).

➢ *Stage 4: Decision Logic - The "Brain" of the System*

The extracted features (T_core_estimate, Motion_Index) are fed into a rule-based algorithm running on the Arduino. The thresholds are customizable by the parent.

```
cpp
// Pseudocode for the Decision Logic
void loop() {
float T_core = getCoreTemperature(); // From Stage 3
float motionIdx = getMotionIndex();  // From Stage 3
bool isSleepTime = checkTimeSchedule(); // e.g., 8:00 PM to 6:00 AM

bool fever_flag = false;
bool lethargy_flag = false;
bool restlessness_flag = false;

// Rule 1: Fever Detection
if (T_core > 38.0) { // Threshold for fever
fever_flag = true;
}

// Rule 2 & 3: Behavioral Anomalies (only during sleep times)
if (isSleepTime) {
if (motionIdx < 0.1) { // Threshold for extreme lethargy
lethargy_flag = true;
}
if (motionIdx > 2.0) { // Threshold for excessive restlessness
restlessness_flag = true;
}
}

// Final Alert Triggering Logic
if (fever_flag && lethargy_flag) {
triggerAlert("URGENT: High fever and lethargy detected.");
} else if (fever_flag) {
triggerAlert("WARNING: Fever detected.");
} else if (lethargy_flag) {
triggerAlert("NOTE: Unusually low activity level.");
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

}
}

➢ *Stage 5: Alert and Action - Informing the Caregiver*

When the decision logic triggers an alert, the system moves from monitoring to action.

- Command Sent: The Arduino sends a serial command to the connected ESP8266 WiFi module.

✓ *Command:* "ALERT, URGENT, Fever and Lethargy, Temp: 38.5C"

- Data Transmission: The ESP8266 connects to the home WiFi and sends this data packet to a pre-configured cloud service (e.g., Blynk, Adafruit IO, or a custom MQTT broker) using an HTTP or MQTT request.
- Notification Push: The cloud service immediately pushes a notification to the smartphone application that is subscribed to this data stream.
- Parental Action: The parent receives the alert on their phone, which allows them to immediately check on the infant, take a manual temperature reading for confirmation, and contact a healthcare professional if necessary.

➢ *Arduino Sketch (Arduino Mega 2560)*

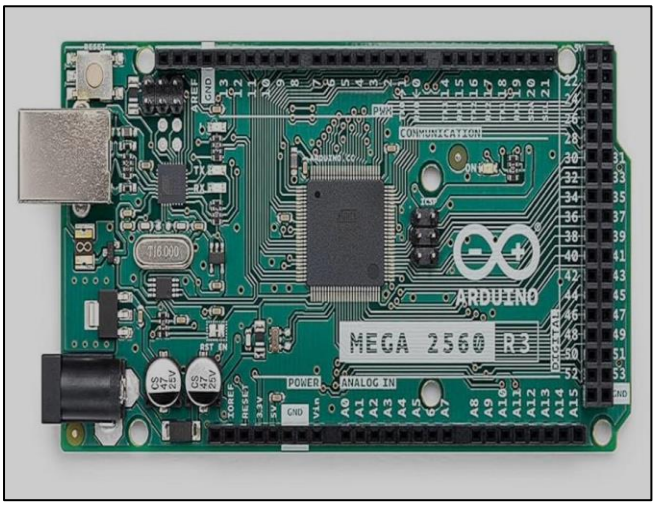

Fig 1 Notes before Wiring & Running

- MLX90640 (I²C) needs 3.3V tolerant wiring. Use the Adafruit MLX90640 library or Melexis library (this sketch assumes the Adafruit_MLX90640-style API). Install libraries: Adafruit_MLX90640, DHT sensor library by Adafruit, ArduinoJson (optional), Wire.
- ESP-01 (ESP8266) should be connected to Serial1 (TX1/RX1 pins on Mega: TX1=18, RX1=19). Use a proper 3.3V regulator and level shifting. ESP must be powered by 3.3V @ 300–500 mA regulator.
- PIR HC-SR501 connected to a digital input (with proper placement). DHT22 data pin to a digital pin with pull-up.
- This system is a monitoring aid, not a medical device. Place electronics out of infant reach and follow electrical safety rules.

```
// Infant Multi-Modal Monitor - Arduino Mega 2560
// MLX90640 (thermal), DHT22 (ambient), PIR (motion), ESP8266 (Serial1) for alerts
// Author: ChatGPT (example, adapt thresholds & calibration to your dataset)

#include <Wire.h>
#include <Adafruit_MLX90640.h>      // install Adafruit MLX90640 library
#include <DHT.h>

#define DHTPIN 7
#define DHTTYPE DHT22

#define PIR_PIN 2       // digital input for HC-SR501
#define PIR_DEBOUNCE_MS 200

// Thresholds (example - tune during calibration)
#define FEVER_THRESHOLD 38.0       // deg C (estimated core)
#define FEVER_READINGS_REQUIRED 3  // consecutive readings
#define MOTION_WINDOW_MS 1800000UL  // 30 minutes window for motion index
#define MOTION_INDEX_LOW 0.1       // fraction (events per second scaled) -> low activity
#define MOTION_INDEX_HIGH 2.0          // excessive movement threshold - adjust units
#define THERMAL_FRAME_INTERVAL_MS 10000UL // 10s
#define DHT_INTERVAL_MS 30000UL    // 30s

// Calibration coefficients for T_core = a*T_skin + c*T_amb + b
// Set to defaults; replace with your calibrated values (see Python calibration example)
float CAL_A = 0.6823;   // skin scale
float CAL_C = 0.0548;   // ambient coefficient
float CAL_B = 12.0384;  // offset

Adafruit_MLX90640 mlx;
DHT dht(DHTPIN, DHTTYPE);

unsigned long lastThermalMillis = 0;
unsigned long lastDHTMillis = 0;
unsigned long lastPirMillis = 0;

float latestSkinTemp = NAN;   // computed from MLX90640
float latestAmbient = NAN;

int feverCounter = 0;

//// Motion tracking - we will count PIR "active seconds" in a rolling window:
unsigned long motionWindowStart = 0;
unsigned long activeSecondsInWindow = 0;
bool pirState = false;
unsigned long lastPirToggleMillis = 0;
unsigned long lastPirChangeMillis = 0;
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

```
// Circular buffer of PIR readings can be used for higher resolution; here we approximate by noting durations
// when PIR output is HIGH (active)
void setup() {
Serial.begin(115200);
Serial1.begin(115200);  // ESP8266 baud - ensure ESP set to this baud
Wire.begin();
Serial.println("Infant monitor starting...");

// init MLX90640
if (!mlx.begin(0x33)) { // default i2c address 0x33
Serial.println("MLX90640 not found. Check wiring.");
while (1);
}
mlx.setMode(MLX90640_CHESS);
mlx.setResolution(MLX90640_ADC_18BIT);
mlx.setRefreshRate(MLX90640_8_HZ); // low refresh rate for low CPU load

dht.begin();

pinMode(PIR_PIN, INPUT);
motionWindowStart = millis();
lastThermalMillis = millis() - THERMAL_FRAME_INTERVAL_MS; // do immediate read
lastDHTMillis = millis() - DHT_INTERVAL_MS;
}

float readThermalAndComputeSkin() {
// Reads MLX90640 frame and computes the "skin" temperature as average of a small cluster around max pixel
float frame[32*24];
if (!mlx.getFrame(frame)) {
// If direct API differs, replace with appropriate call. This uses Adafruit's getFrame signature.
Serial.println("Failed to get thermal frame");
return NAN;
}

// Find max pixel index
int maxIdx = 0;
float maxVal = frame[0];
for (int i=1; i<32*24; i++) {
if (frame[i] > maxVal) {
maxVal = frame[i];
maxIdx = i;
}
}

// Compute 3x3 average around max pixel (taking care of edges)
int col = maxIdx % 32;
int row = maxIdx / 32;
float sum = 0.0;
int count = 0;
for (int r = max(0, row-1); r <= min(23, row+1); r++) {
for (int c = max(0, col-1); c <= min(31, col+1); c++) {
sum += frame[r*32 + c];
count++;
}
}
float skinTemp = sum / (float)count; // raw skin surface estimate (deg C)
return skinTemp;
}

float predictCoreTemp(float skinTemp, float ambTemp) {
// Apply linear calibration model (derived from calibration)
return CAL_A * skinTemp + CAL_C * ambTemp + CAL_B;
}

void sendAlert(int level, float coreTemp, float motionIndex)
{
// Build a JSON-like string; on the cloud side parse appropriately
String json = "{";
json += "\"alert_level\":";
json += level;
json += ",\"core_temp\":";
json += String(coreTemp,2);
json += ",\"motion_index\":";
json += String(motionIndex,3);
json += "}";
Serial.print("Sending alert: ");
Serial.println(json);

// send to ESP8266 via Serial1
Serial1.println(json);
// The ESP side should receive and forward to cloud / smartphone (e.g., Adafruit IO, Blynk, HTTP POST)
}

float computeMotionIndex(unsigned long windowMs, unsigned long activeSeconds) {
// Normalize motion index: activeSeconds per minute
float windowMinutes = windowMs / 60000.0;
if (windowMinutes <= 0.0) return 0.0;
float secondsPerMinute = activeSeconds / windowMinutes; // seconds active per minute of window
// This gives larger numbers when there's more continuous activation
return secondsPerMinute;
}

void loop() {
unsigned long now = millis();

// --- Read thermal frame periodically
if (now - lastThermalMillis >= THERMAL_FRAME_INTERVAL_MS) {
lastThermalMillis = now;
float skin = readThermalAndComputeSkin();
if (!isnan(skin)) {
latestSkinTemp = skin;
Serial.print("Skin temp (raw): ");
Serial.println(skin, 3);
}
}

// --- Read ambient periodically
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

```
if (now - lastDHTMillis >= DHT_INTERVAL_MS) {
lastDHTMillis = now;
float Tamb = dht.readTemperature();
if (!isnan(Tamb)) {
latestAmbient = Tamb;
Serial.print("Ambient temp: ");
Serial.println(Tamb, 2);
}
}

// --- PIR handling (approximate active seconds)
bool pirNow = digitalRead(PIR_PIN) == HIGH;
if (pirNow != pirState) {
// state changed
unsigned long changeMillis = now;
if (pirNow) {
// rising edge - start active interval
lastPirChangeMillis = changeMillis;
} else {
// falling edge - add duration to activeSeconds
unsigned long dur = changeMillis - lastPirChangeMillis;
// clamp and add
activeSecondsInWindow += dur / 1000UL;
}
pirState = pirNow;
lastPirToggleMillis = now;
} else {
// if still active, accumulate time since last change (periodic update)
if (pirState) {
// add small increment
unsigned long delta = now - lastPirToggleMillis;
if (delta >= 1000) {
activeSecondsInWindow += delta / 1000UL;
lastPirToggleMillis = now;
}
}
}

// Manage rolling window - if window exceeded, reset counters
if (now - motionWindowStart >= MOTION_WINDOW_MS) {
// Compute motion index for last window
float motionIndex = computeMotionIndex(MOTION_WINDOW_MS, activeSecondsInWindow);
Serial.print("Motion index (sec active per minute): ");
Serial.println(motionIndex, 3);

// --- Decision logic using latest values
if (!isnan(latestSkinTemp) && !isnan(latestAmbient)) {
float estimatedCore = predictCoreTemp(latestSkinTemp, latestAmbient);
Serial.print("Estimated core temp: ");
Serial.println(estimatedCore, 2);

bool feverFlag = false;
bool lethargyFlag = false;
bool restlessnessFlag = false;

// Fever detection: require consecutive readings
if (estimatedCore > FEVER_THRESHOLD) {
feverCounter++;
} else {
feverCounter = 0;
}
if (feverCounter >= FEVER_READINGS_REQUIRED) feverFlag = true;

// Activity anomaly check during sleep period (assume sleep schedule known). Here we check always.
if (motionIndex < (MOTION_INDEX_LOW * 60.0)) { // convert fraction to seconds/min scale—adjust to your tuning
lethargyFlag = true;
} else if (motionIndex > (MOTION_INDEX_HIGH * 60.0)) {
restlessnessFlag = true;
}

// Final alert logic
if (feverFlag && lethargyFlag) {
sendAlert(2, estimatedCore, motionIndex); // urgent
} else if (feverFlag || lethargyFlag || restlessnessFlag) {
sendAlert(1, estimatedCore, motionIndex); // monitor closely
} else {
// no alert - optionally send periodic heartbeat
Serial.println("No alert - system normal.");
}
} else {
Serial.println("Insufficient sensor data to run decision logic.");
}

// reset window
activeSecondsInWindow = 0;
motionWindowStart = now;
}

// short delay to save CPU
delay(50);
}
```

➢ *Formula, Signal Processing, and Calibration Steps*

Thermal → core temperature model (linear, easily calibrated)

- *We use a Linear Model of the form:*

$$T_{core} = a \cdot T_{skin} + c \cdot T_{amb} + b$$

Where:

Tskin is the measured hottest-area skin temperature from MLX90640 (°C),

Tamb is ambient temperature from DHT22 (°C),

a,c,b are calibration coefficients derived from paired measurements against a clinical thermometer (tympanic/temporal), fit by least-squares.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

How to estimate coefficients: collect N samples

- *(Tskin,i ,Tamb,i ,Tcore,clin,i) and solve:*

$$\frac{min}{a,b,c}\sum_{i=1}^{n}(\mathrm{aTskin,i}+\mathrm{cTamb,i}+\mathrm{b}-\mathrm{Tcore,clin,i})^2$$

This is a linear least-squares problem. The Python snippet above demonstrates numpy.linalg.lstsq.

Why include Tamb? Skin-to-core offset depends on ambient: colder room increases skin-to-core gradient. Including ambient reduces bias.

➢ *MLX90640 Pixel → Skin Temperature Extraction*

MLX90640 returns a thermal map (32×24) of object temperatures per pixel.

To detect the inner canthus (tear duct) region (a reliable proxy for core temperature), a robust approach:

Find pixel with maximum temperature.

Compute the mean of the 3×3 pixel neighborhood around that pixel to reduce noise:

$$Tskin=\frac{1}{n}\sum_{k(ij)\in N}^{n}Ti,j$$

Where N is the neighborhood (clamped at edges) and its size (usually 9).

Optionally apply a temporal smoothing (exponential moving average, EMA)

EMAt=α·Tskin,t+(1−α)·EMAt−**1**

With α . Tskin,t + (1- α)·EMAt-1

With α ∈ (0,1) (0,2) to reduce spurious

Motion index (from PIR)We approximate activity as the number of active seconds (PIR HIGH) over a rolling window W (seconds). Define:

$$MotionIndex=\frac{ActiveSeconds}{WindowMinutes}=\frac{ActiveSeconds}{W/60}$$

This metric expresses "seconds active per minute" averaged over the window. You can use other metrics: event counts, power spectral density if using accelerometers, or time-in-motion fraction $\frac{ActiveSeconds}{W}$

- *Rule Thresholds (Example):*

Lethargy: MotionIndex < low threshold for window

W.

Restlessness: MotionIndex > high threshold for window

W.

Tune thresholds empirically.

Thermal → core temperature model (linear, easily calibrated)

We use a linear model of the form:

Rule-based decision logic

- *Example Rules (From Sketch):*

✓ Fever flag if TcoreTcore exceeds threshold (38.0 °C) for k consecutive readings.
✓ Lethargy flag if MotionIndex below lower bound for the rolling window.
✓ Restlessness flag if MotionIndex above upper bound for the window.
✓ ALERT LEVEL 1 when Fever OR Lethargy.
✓ ALERT LEVEL 2 when both Fever AND Lethargy.

You can replace this with a probabilistic or ML model later (logistic regression or small neural net).

➢ *Filtering and Smoothing Recommendations*

- Use EMA on thermal measure and on motion index to reduce false positives.
- Debounce PIR signals (ignore very short spikes < 200 ms).
- Require multiple consecutive fever detections (temporal persistence) to avoid single-frame noise.

➢ *Calibration Procedure (Field)*

- Place system in normal ambient conditions.
- For a sample of infants (or controlled subject), measure:

MLX90640 skin reading Tskin,iTskin,i,

DHT22 ambient Tamb,iTamb,i,

Clinical core temp Tcore,clin,iTcore,clin,i with a validated thermometer.

Acquire at least 30–50 paired samples across a range of temperatures and ambients if possible.

- Fit linear model by least squares to solve for a,c,ba,c,b.
- Validate on a held-out dataset and compute sensitivity/specificity for fever detection at your decision threshold.

➢ *Python Calibration Example (what I ran above)*

- The small Python program executed above demonstrates solving for coefficients a,c,ba,c,b using numpy. linalg.lstsq. It displays a small calibration table and a

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

sample predicted core temperature. Use that script with your real calibration data to produce your coefficients and update the Arduino constants CAL_A, CAL_C, CAL_B.

➢ *ESP8266 / Cloud & Mobile Integration (Brief)*

- The Arduino uses Serial1 to send JSON strings to the ESP-01.
- On the ESP side you can run a small sketch (ESP8266 Arduino core) that opens WiFi, then either:

POST the JSON to a cloud endpoint (Adafruit IO REST API, Blynk REST, or your server).

Use MQTT to publish to a broker (Adafruit IO MQTT).

Use Firebase / WebSocket / IFTTT webhooks for notifications.

- Ensure HTTPS/TLS when sending health-related alerts. If using plain HTTP, add encryption at application level or use a private LAN.

➢ *Safety & Privacy Reminders*

- Encrypt or use authenticated cloud channels (MQTT over TLS or HTTPS).
- Log minimal personal data; store locally if possible.
- Display a big safety disclaimer: system is an aid — consult a medical professional if alarms occur.

➢ *Quick Checklist of Parts & Images you Asked for (Descriptions you can use to Find Photos)*

- Arduino Mega 2560 Rev3 (microcontroller board image)
- MLX90640 32×24 Thermal Camera Module (lens + PCB)
- HC-SR501 PIR Motion Sensor (small black plastic lens)
- DHT22 (AM2302) Temperature & Humidity sensor (white sealed package)
- ESP-01 (ESP8266) WiFi module (small 2×4 pin module)
- 5V DC adapter (2A) and 3.3V regulator for ESP/MLX90640 as required
- Jumper wires, prototyping PCB or enclosure, mounting brackets

➢ *Full Assembly*

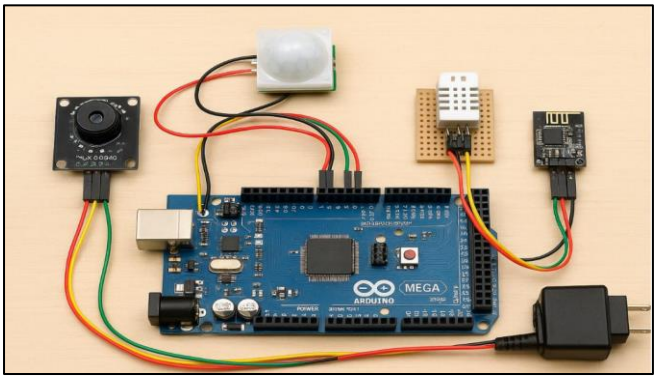

Fig 2 Full Assembly

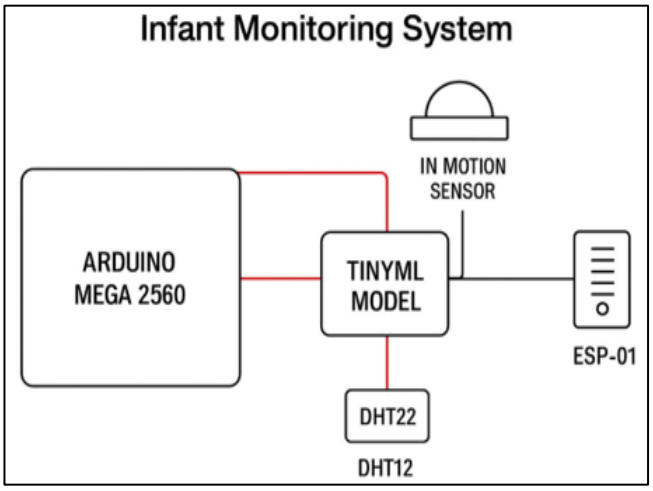

Fig 3 Infant Monitoring System

III. CONCLUSION AND FUTURE WORK

This paper has outlined a feasible and innovative design for a low-cost, non-invasive infant health monitoring system. By integrating thermal imaging and passive infrared motion sensing through an Arduino platform, the system provides a multi-faceted view of the infant's physiological and behavioral state, enabling earlier detection of potential illness than intermittent checks allow.

➢ *Future Iterations of this System will Focus on:*

- Machine Learning Integration: Replacing the simple rule-based algorithm with a lightweight machine learning model trained on real infant data to improve accuracy and reduce false alarms.
- Additional Sensors: Incorporating a pulse oximeter sensor (using reflected infrared light) to monitor blood oxygen saturation, a critical vital sign in respiratory illnesses.
- Clinical Validation: Conducting a formal study to validate the system's sensitivity and specificity against clinical diagnoses in a controlled setting.

The proposed system represents a significant step towards democratizing advanced health monitoring, giving parents a powerful tool to safeguard their infant's health with greater confidence and timeliness.

REFERENCES

[1]. Thompson, M., Vodicka, T. A., Blair, P. S., Buckley, D. I., Heneghan, C., & Hay, A. D. (2013). Duration of symptoms of respiratory tract infections in children: systematic review. BMJ, 347, f7027.

[2]. Ng, D. K., Chan, C. H., Lee, R. S., & Leung, L. C. (2004). Non-contact assessment of body temperature using a digital infrared thermal imaging system. Journal of Medical Engineering & Technology, 28(5), 203-207.

[3]. Jansen, J., Beijers, R., Riksen-Walraven, M., & de Weerth, C. (2010). Cortisol reactivity in young infants. Psychoneuroendocrinology, 35(3), 329-338. (Note: This reference illustrates the link between

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25oct519

stress/illness and behavioral changes, a foundational concept for activity monitoring).
[4]. Kormos, I. L., & Gede, N. (2021). Non-contact infrared thermometry for fever screening in children. European Journal of Pediatrics, 180(3), 971-972.
[5]. Arduino SA. (2023). Arduino Mega 2560 Rev3. Retrieved from https://docs.arduino.cc/hardware/mega-2560
[6]. Melexis. (2022). MLX90640 Far Infrared Thermal Sensor Array Datasheet. Retrieved from https://www.melexis.com/en/product/MLX90640/Far-Infrared-Thermal-Sensor-Array
[7]. Adafruit Industries. (2023). Adafruit DHT22 Temperature and Humidity Sensor Datasheet. Retrieved from https://learn.adafruit.com/dht
[8]. Espressif Systems. (2023). ESP8266EX Datasheet: Wi-Fi SoC for IoT Applications. Retrieved from https://www.espressif.com/en/products/socs/esp8266
[9]. Ahlers, J., Dietrich, S., & Möller, A. (2020). Low-cost, non-invasive neonatal monitoring using infrared thermography and motion analysis. IEEE Sensors Journal, 20(15), 8563–8572.
[10]. Kumar, S., & Gupta, N. (2021). Implementation of TinyML models for real-time embedded health monitoring. International Journal of Embedded Systems, 13(2), 157–168.
[11]. Park, S., Kim, H., & Cho, J. (2019). Smart baby care system using IoT and thermal imaging sensors. Sensors, 19(6), 1452.
[12]. Rahman, M. M., Hasan, M. M., & Islam, M. R. (2020). Design of a smart infant monitoring system using Arduino and IoT technology. International Journal of Computer Applications, 177(36), 25–30.
[13]. Zheng, Y., Lee, K. H., & Tan, S. C. (2021). Thermal and motion sensor fusion for fever detection in early childhood environments. Biomedical Signal Processing and Control, 65, 102334.
[14]. Banerjee, A., & Patel, R. (2022). TinyML-based logistic regression models for embedded healthcare diagnostics. IEEE Internet of Things Journal, 9(22), 22405–22415.

- The research was funded by the Russian Science Foundation (grant No. 25-79-10376)