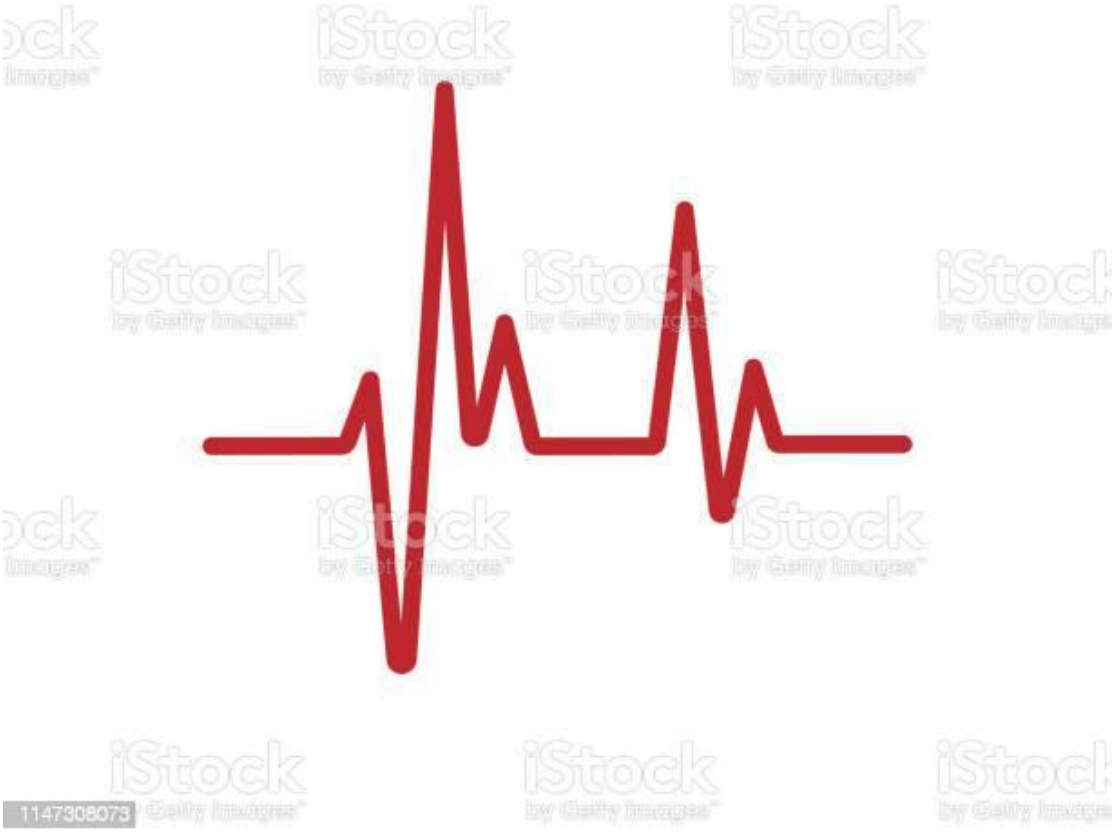


Final Project Report

CENG 317 Hardware Production Tech

Team name: **BioBytes**

Project name: **Accelerometer in Safe Infant Health Monitor (Sihmon)**

Student Name: **Naman Pal**

Student ID: N01458339

Date of submission: August 16, 2023

Declaration of Authorship

I, Naman Pal, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included. The contributions from the team members are as follows: -

Scrum Master: Naman Pal

Web interface and database incharge: Eshan Salwan

App functionality incharge: Zoyeba Mahbub

Debugging and refactoring incharge: Satinder Kaur

Project Abstract

This project endeavours to create an innovative infant bracelet endowed with the capability to comprehensively monitor vital health parameters, including heart rate, oxygen saturation, body temperature, motion dynamics, and ambient noise levels. Through seamless integration with an Android application, this bracelet offers caregivers real-time access to a wealth of health data, enabling them to make informed decisions regarding the well-being of their infants.

Central to the functionality of the bracelet is the meticulous incorporation of the ADXL345 triple axis accelerometer sensor, a pivotal component managed by the project's scrum master. By adeptly capturing acceleration measurements along the x, y, and z axes, this sensor forms the bedrock for acquiring intricate motion-related insights pivotal for understanding an infant's activity trends.

The amassed data undergoes seamless transmission via a Raspberry Pi, which serves as a conduit directing the information flow to an online repository hosted on the reliable Google Firebase platform. This repository serves as a resilient storehouse, facilitating the efficient archival and retrieval of the vital health metrics data. Subsequently, the Android application processes the data using advanced algorithms, furnishing caregivers with real-time updates on their infant's health status.

The project holds particular significance for caregivers tending to infants susceptible to health conditions such as Sudden Infant Death Syndrome (SIDS). By ensuring constant health surveillance and immediate alerts, the project emerges as a preventive measure, mitigating potential risks and contributing to the safety of infants. Beyond infancy, the project envisions expansion into a broader patient demographic, thereby advancing the goal of fostering a healthier society. In sum, this endeavour harmonises cutting-edge technology with healthcare, fostering a future where well-informed caregiving and advanced monitoring converge for the greater good.

Table of Contents

Introduction To Our Project

In the midst of a rapidly evolving technological landscape, this project emerges as a beacon of innovation in the realm of infant care, poised to redefine how caregivers engage with monitoring and ensuring the well-being of their infants. At its heart, this endeavour is focused on introducing a groundbreaking infant monitoring system that harmoniously combines the capabilities of a state-of-the-art wearable bracelet with the prowess of an advanced Android application. Through this harmonious integration, caregivers are equipped with a comprehensive and real-time health tracking solution that transcends conventional methods.

Central to the project's aspiration is the creation of an intelligent and perceptive infant bracelet, equipped with the ability to meticulously track an array of vital health metrics. This encompasses critical parameters like heart rate, oxygen saturation, body temperature, motion dynamics, and ambient noise levels. The intrinsic connection to an Android application heralds a paradigm shift in how caregivers access, interpret, and act upon the infant's health-related data.

An essential linchpin of this bracelet's functionality is the seamless integration of the ADXL345 triple axis accelerometer sensor, a meticulous undertaking led by the project's adept scrum master. By precisely capturing acceleration data across multiple axes, this sensor lays the foundation for extracting nuanced insights into the infant's movements and activities, allowing for a comprehensive understanding of their well-being.

This data is transmitted via a Raspberry Pi to a secure online repository hosted on the robust Google Firebase platform. Serving as a fortified vault, this repository efficiently organises and safeguards the stream of invaluable health metrics. The Android application emerges as the data maestro, employing advanced algorithms to process this influx of information, presenting caregivers with immediate, real-time updates and in-depth analyses. The Android App is hosted on Google Play, and has a simple to use UI, which is great for any parent/caregiver, no matter their technical knowledge.

While the project's core commitment lies in safeguarding vulnerable infants, particularly from the risks associated with Sudden Infant Death Syndrome (SIDS), its aspirations extend far beyond infancy. The Android application's potential for a diverse patient demographic underscores its transformative role in shaping holistic health monitoring practices across society. Thus, the project embodies a visionary fusion of cutting-edge Android application technology with empathetic caregiving, paving the way for an era where innovation converges with compassion to forge new standards of infant and patient care.

Project Proposal

Project Title and Team Members:

Project Title: Smart Infant Health Monitor (Sihmon)

Team Members:

Naman Pal: n01458339
Eshan Salwan: n01422232
Satinder Kaur: n01421666
Zoyeba Mahbub: n01445837

Capstone Team Setup:

We have formed a dedicated capstone team within the CENG 322 class, consisting of members who previously attended the same professor's class in CENG 317. This familiarity with the professor's teaching style and expectations will aid in smooth collaboration and project execution.

Project Summary and Background:

Our project, the SmartBaby Health Monitor, focuses on developing an innovative infant health monitoring system that combines a wearable smart bracelet with an advanced Android application. The background of our project lies in addressing the need for real-time health tracking solutions for infants, aiming to provide caregivers with actionable insights for enhanced infant care practices.

Technical Implementation Plan:

We plan to implement the hardware aspect of our project by designing and assembling a printed circuit board (PCB) that incorporates various sensors for monitoring vital health metrics like heart rate, oxygen saturation, body temperature, motion dynamics, and ambient noise levels. The data collected by these sensors will be processed and transmitted to an Android application for analysis and visualisation.

Data Source:

The data for our project will be sourced from the sensors embedded in the wearable smart bracelet. These sensors will directly capture the health metrics of the infant.

PCB Assembly Option:

We intend to opt for Option1, Ordered it online, soldered in the Humber prototype lab.

Time Estimation and Contingency Plan:

We estimate that the project will be completed within 2 months. To ensure timely completion, we will maintain a detailed project timeline with milestones and checkpoints. In case of any unexpected challenges, we have allocated buffer time to address setbacks and remain on track. Regular team meetings and progress evaluations will contribute to effective project management.

Market Research and Differentiation:

We conducted research on similar products in the market and found no such product. There are fitbit watches and the like, but our project targets a completely different issue. Our Smart Infant Health Monitor differentiates itself by offering a comprehensive health monitoring solution specifically tailored for infants. Unlike existing products, our project aims to integrate advanced sensor technologies with a user-friendly Android application, providing caregivers with real-time insights and alerts for proactive infant care.

Conclusion:

The SmartBaby Health Monitor project presents an exciting opportunity to contribute to infant care and well-being through the integration of innovative hardware and software solutions. By bridging sensor technology and mobile app development, we aspire to redefine how caregivers monitor and ensure the health of infants, fostering a safer and healthier environment for the youngest members of our society.

References:

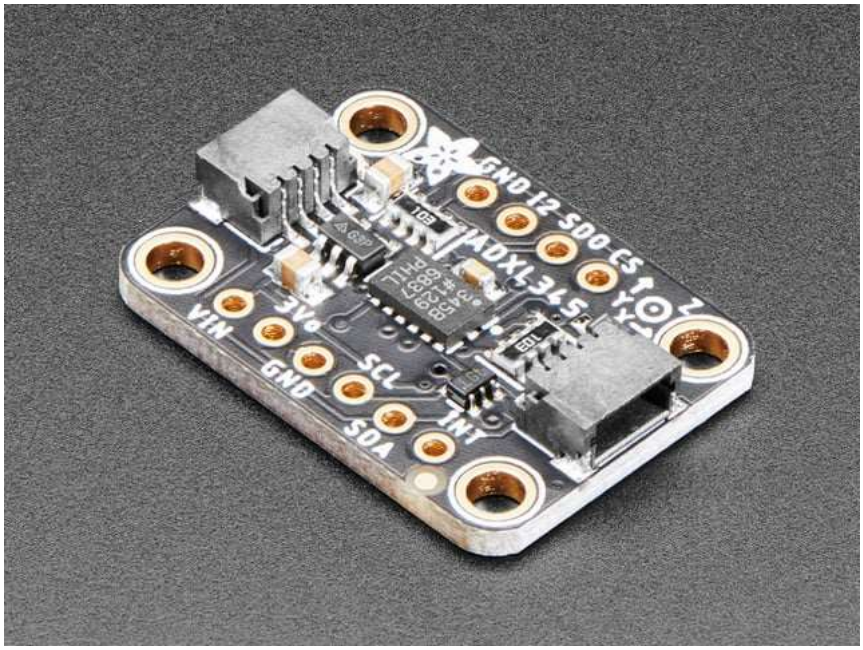
https://www.parents.com/baby/health/sids/does-your-baby-need-a-sids-monitor/

Sensor Details

Sensor: Accelerometer sensor for measuring shivers
Model: ADXL 345 3-axis accelerometer sensor

Datasheet

(The next 2 pages are the datasheet of the ADXL 345 breakout board)



ADXL345 Digital Accelerometer

By Bill Earl
Triple-axis accelerometer breakout with a digital interface.

Feedback? Corrections?

Pinouts

Save Subscribe
×

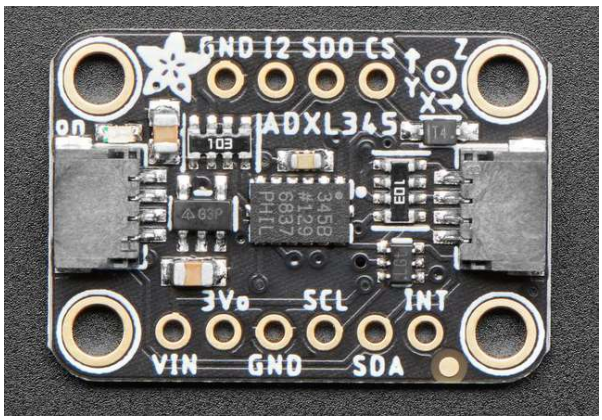
New Subscription

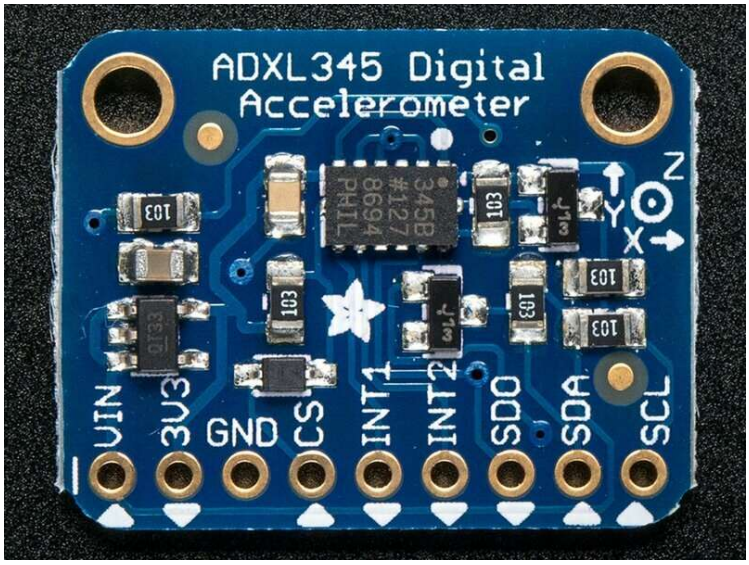
Please sign in to subscribe to this guide.

You will be redirected back to this guide once you sign in, and can then subscribe to this guide.

Close

The ADXL345 breakout has the following pinout:





The default I2C address for this board is **0x53**.

Power Pins

This breakout board can be run on **3.3V** and **5V** systems. We added an on-board 3.3V regulator and logic-level shifting circuitry, making it a perfect choice for interfacing with any 3V or 5V microcontroller such as the Arduino.

- **VIN -** This is the input to the 3.3V voltage regulator, which makes it possible to use the 3.3V sensor on 5V systems. It also determines the logic level of the SCL and SDA pins. Connect this to **3.3V** on the MCU for 3.3V boards (Adafruit Feathers), or **5.0V** for 5V Arduinos (Arduino Uno, etc.).
- **3Vo -** This is the **OUTPUT** of the 3.3V regulator, and can be used to provide 3.3V power to other parts of your project if required (< 100mA).
- **GND -** Connect this to the **GND** pin on your development board to make sure they are sharing a common GND connection, or the electrons won't have anywhere to flow!

I2C Pins

- **SCL -** The clock line on the I2C bus. This pin has an internal pullup resistor on the PCB, which is required as part of the I2C spec, meaning you don't need to add one externally yourself. This also functions as **SCK** in SPI mode.
- **SDA -** The data line on the I2C bus. This pin has an internal pullup resistor on the PCB, which is required as part of the I2C spec, meaning you don't need to add one externally yourself. This also functions as **MOSI** in SPI mode.
- **STEMMA QT -** These connectors allow you to connect to development boards with **STEMMA QT** connectors, or to other things, with various associated accessories.

The available pins are identical for both versions of the boards. However, on the STEMMA QT version, pins I2 (INT2), SDO, and CS are located at the top of the board.

Other Pins

- **SDO/ALT ADDR -** This pin can be used as **MISO** in SPI mode, but is more commonly used as an optional bit in the I2C bus address. By default this pin is pulled down, meaning it has a value of **0** at startup, which will results in an I2C address of **0x53**. If you set this pin high (to 3.3V), and reset, the I2C address will be updated to **0x1D**.
- **CS**: This dual purpose pin can be used as the chip select line in SPI mode, but also determines whether the board will boot up into I2C or SPI mode. The default of logic high sets the board up for I2C, and manually setting this pin low and resetting will cause the device to enter SPI mode. Please note that SPI mode is not actively support and the SPI pins are not all 5V safe and level shifted, so care will be required when using it!
- **INT1** and **INT2**: There are two optional interrupt output pins on this sensor, which can be configured to change their state when one or more 'events' occur. For details on how to use these interrupts, see the Arduino/HW Interrupts page later in this guide.

LED Jumper

- **LED jumper -** This jumper is located on the back of the board. Cut the trace on this jumper to cut power to the "on" LED.

Overview Assembly and Wiring

This guide was first published on Mar 26, 2013. It was last updated on Mar 26, 2013.

This page (Pinouts) was last updated on Mar 25, 2022.

Text editor powered by tinymce.
Difficulty: Intermediate
Guide Type: Product
Products:
Contributors: Bill Earl, Liz Clark, Kattni Rembor, Danny Nosonowitz
Categories: Adafruit Products
Breakout Boards/Accel, Gyro, & Magnetometers
STEMMA
30 Saves
Featured Products

Introduction to CPU board

Raspberry Pi Overview:

The Raspberry Pi acts as the central processing unit (CPU) for our SmartBaby Health Monitor project. It executes the code responsible for data collection from the ADXL345 sensor, processes the captured motion dynamics, and communicates with the Android application for real-time health insights. At its core, the Raspberry Pi is a credit-card-sized computer equipped with processing power, memory, and various interfaces, all integrated onto a single board. This small yet capable SBC boasts GPIO (General-Purpose Input/Output) pins that enable connections to external hardware components, making it an ideal platform for developing interactive projects that demand computational capabilities.

ADXL345 Breakout Board Overview:

One of the essential components of our Smart Infant Health Monitor project is the ADXL345 triple axis accelerometer sensor, integrated into our custom-designed PCB board. The ADXL345 sensor captures the acceleration forces experienced by the infant in three dimensions. These acceleration readings are critical for understanding the baby's movements and activities, forming the basis for motion-related health insights provided to caregivers through the Android app. Its compact size, low power consumption, and ease of integration make it a perfect fit for our real-time health monitoring application.
The ADXL345 breakout board comprises the following essential components and pins: -

Mechanical Accelerometer Sensor on the ADXL breakout board:

This sensor is responsible for detecting and measuring acceleration forces in three dimensions – along the x, y, and z axes. It's the heart of our motion monitoring system.

Communication Interface:

The ADXL345 supports communication through I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) protocols. This allows seamless data exchange between the sensor and the Raspberry Pi.

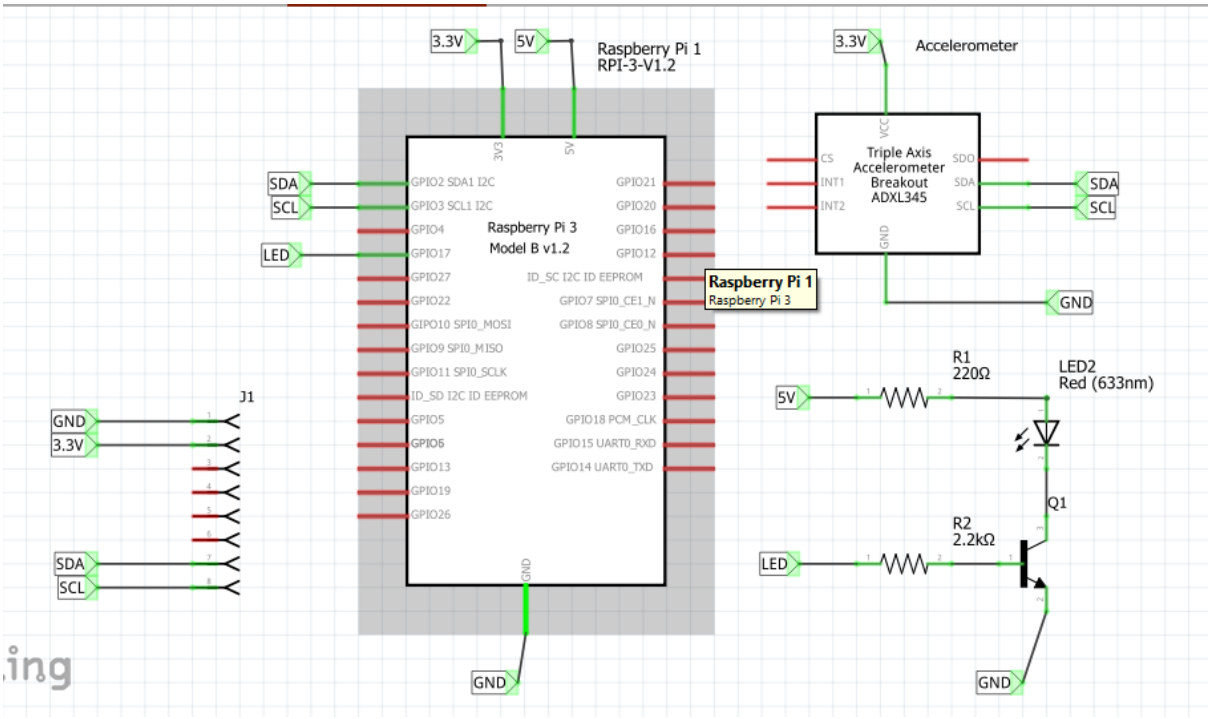
Power Supply Pins:

These pins provide the necessary voltage (3.3V) to power the sensor, ensuring its proper functioning.

Ground Pins:

Ground pins establish a reference point for the electrical circuit, enabling the sensor to communicate accurately with other components.

Schematic Design of Hardware



Components:

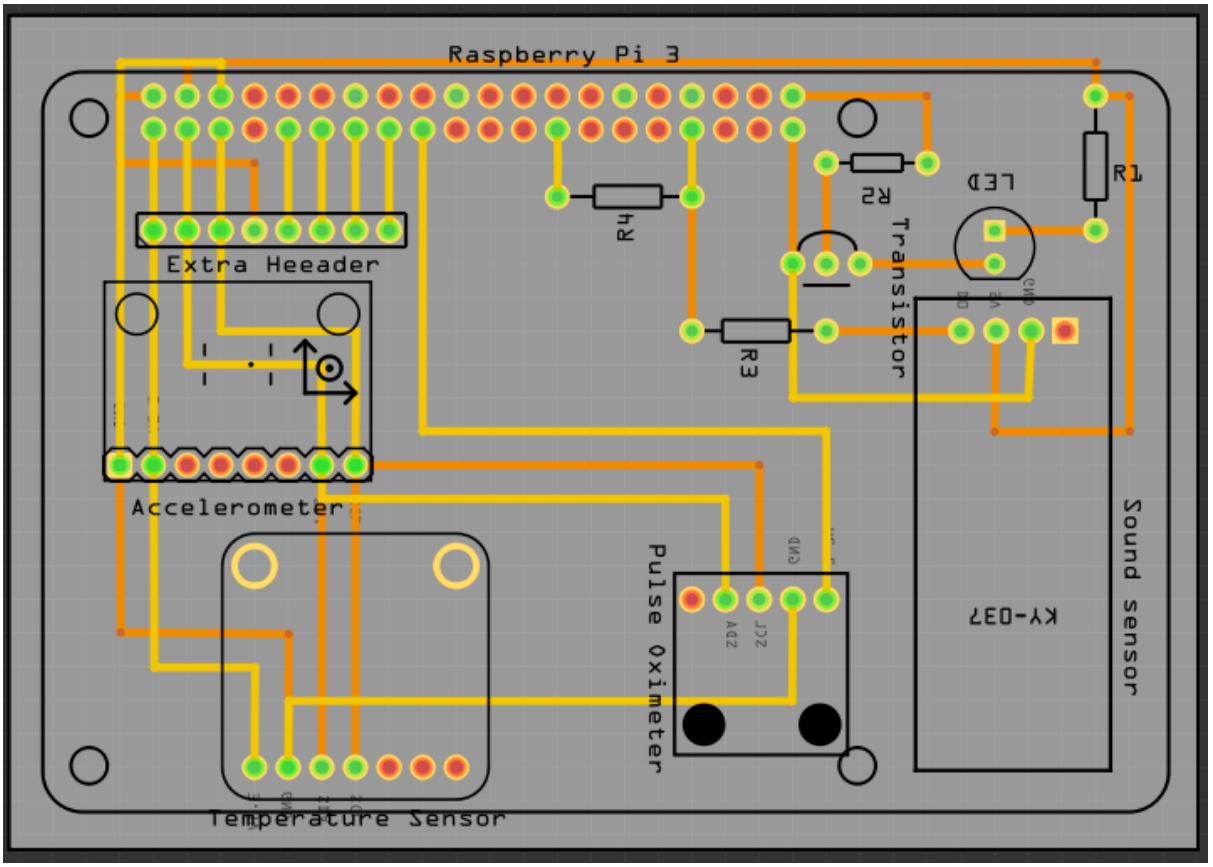
Raspberry Pi 4: A microprocessor board.
LED (Light Emitting Diode): A diode that emits light when current flows through it.
Resistor (220Ω, 2.2kΩ): Limits the current flowing through the LED to prevent damage.
Transistor: to regulate LED circuit
8 pin header: to house the ADXL 345 breakout board
ADXL345 Triple Axis Accelerometer Sensor: A sensor that measures acceleration forces in three dimensions.

The connections can be traced with the help of pin labels.

Functionality:

When the Raspberry Pi 4 is powered on, the program running on it controls the LED via GPIO Pin 17. The high signal from GPIO Pin 17 flows through the 220Ω resistor, lighting up the LED. Simultaneously, the Raspberry Pi 4 communicates with the ADXL345 sensor via the I2C protocol. The ADXL345 sensor measures acceleration forces in three dimensions and sends the data to the Raspberry Pi 4. The Raspberry Pi 4 processes the acceleration data.

PCB Design



PCB Testing Procedure

1) PCB Assembly:

Assemble the components on the PCB according to the schematic design, ensuring correct placement and orientation.
Solder the components onto the PCB, ensuring proper connections and avoiding any solder bridges.

2) Visual Inspection:

Visually inspect the PCB for any soldering errors, short circuits, or misplaced components.
Verify that the connections match the schematic and that no components are installed backward.

3) Power Connection:

Connect a Raspberry Pi (also a power source to the PCB)
Ensure that the power source provides the correct voltage and current for the components.

4) Programming the Raspberry Pi:

Load the python program onto the Raspberry Pi 4. This includes programming to control the LED blinking and communicate with the ADXL345 sensor.
Verify that the software is correctly configured and compatible with the hardware.

5) Initial Testing:

Power on Raspberry Pi, and hence, the PCB .
Observe the LED's behaviour to see if it blinks as intended. It should follow the programmed pattern based on the ADXL345 sensor's data.

6) Data Communication Testing:

Check if the Raspberry Pi can communicate with the ADXL345 sensor through the I2C interface.
Confirm that the Raspberry Pi is receiving acceleration data from the sensor.

7) Interactive Testing:

Physically move or tilt the ADXL345 sensor to generate acceleration data.
Observe how the LED responds to changes in acceleration, as programmed.
If the LED blinking pattern doesn't match expectations, troubleshoot and adjust the software code or sensor calibration.

8) Data Accuracy:

Ensure that the acceleration data received from the ADXL345 sensor aligns with the actual movements or changes in orientation.

9) Troubleshooting:

If there are any issues, carefully review the connections, soldering, and software code.
Check for loose connections, damaged components, or incorrect configurations.

10) Repeat Testing:

Perform multiple testing cycles to ensure consistent and reliable operation of the circuit.

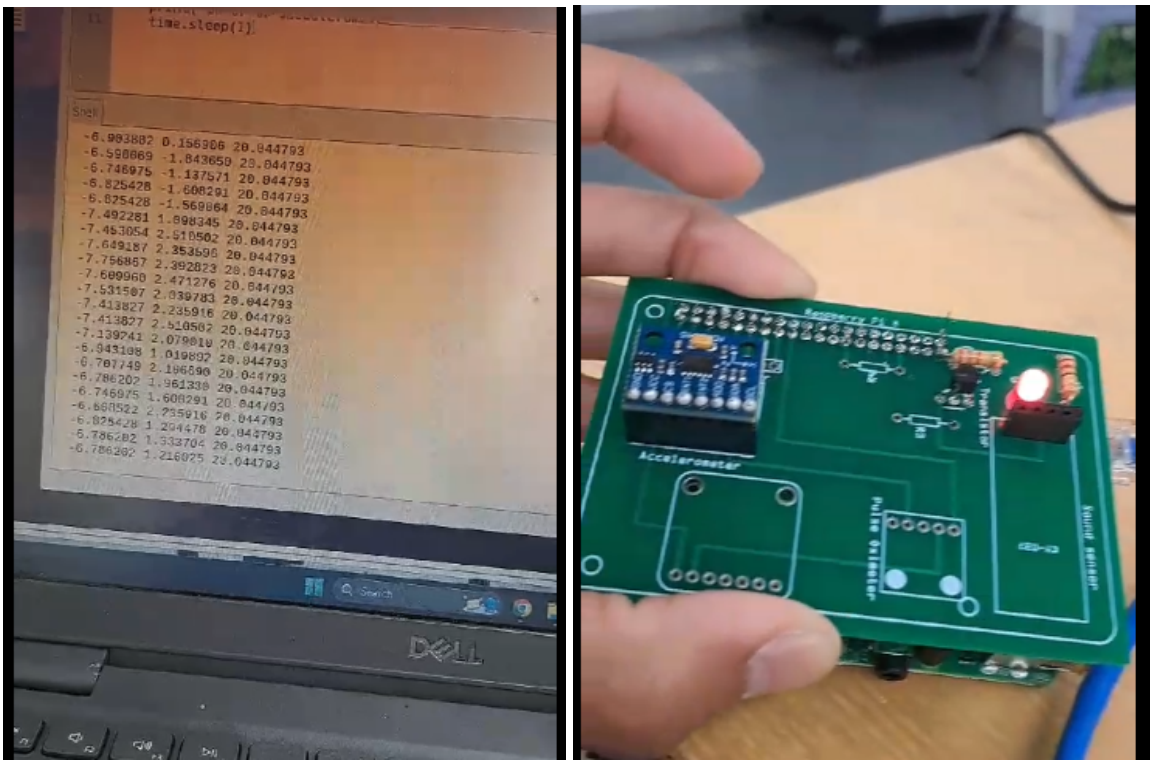
11) Final Validation:

Document the testing results, including any modifications made during troubleshooting.
Confirm that the LED blinking circuit and ADXL345 sensor operate as expected on the PCB.
Remember that PCB testing may involve iterative steps, adjustments, and troubleshooting. Documenting your testing process and any changes made will aid in tracking progress and ensuring a successful outcome.

GitHub Website Link

https://github.com/ZoyebaMahbub5837/InfantHealthMonitor

PCB Testing Results



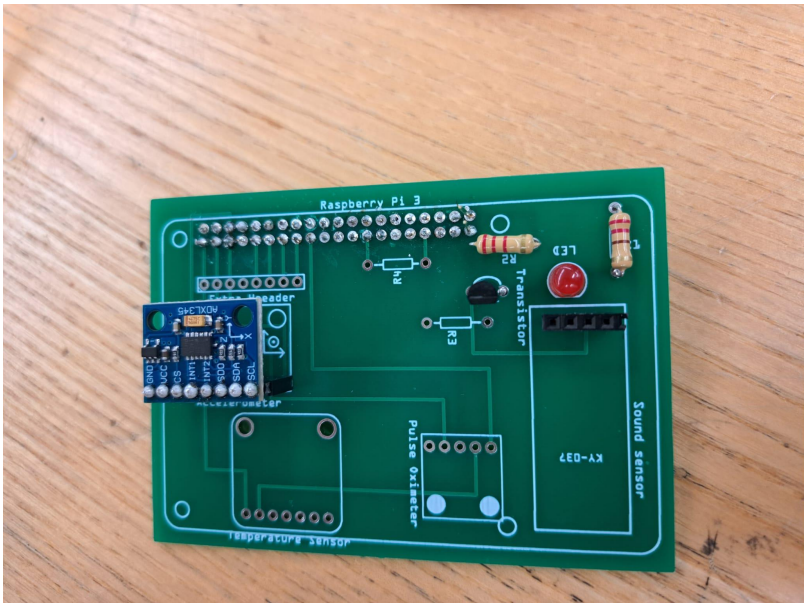
Pictures showing successful PCB testing; Successful readings from sensor (LEFT) LED blinking (Right)

Summary

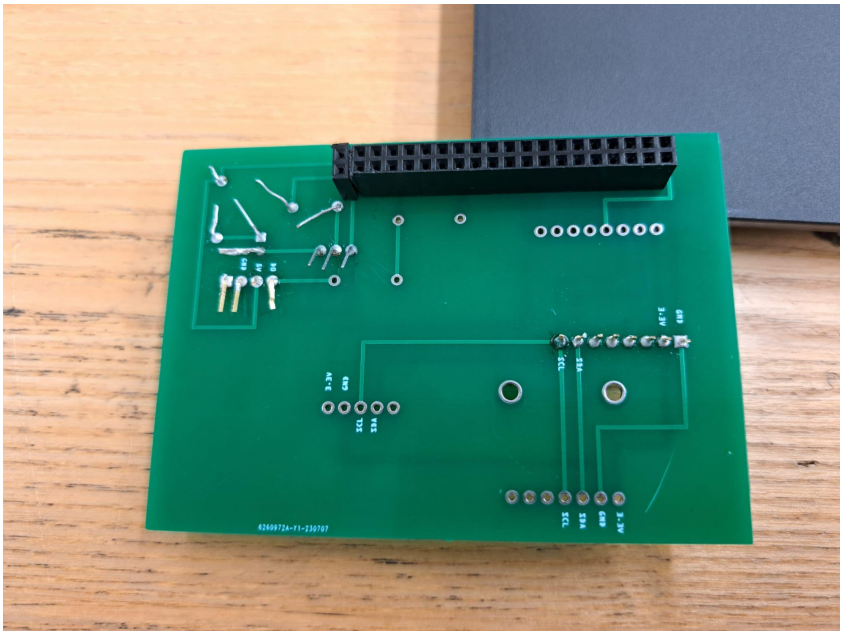
The PCB testing was successful. The code can be found in the appendix. The details of the circuit and the PCB as well as the Schematic can be found above.

Pictures of the assembled PCB

Front



Back



Troubleshooting Procedures

1. Visual Inspection:

Inspect the soldered connections, components, and PCB for any visible solder bridges, cold solder joints, or misplaced components.
Check for any damage to the components, traces, or PCB.

2. Power and Ground:

Ensure that the power and ground connections are correctly established for all components.
Verify that the power supply voltage matches the requirements of the components.

3. Component Orientation:

Double-check that all components, including the LED, ADXL345 sensor, and resistors, are correctly oriented and soldered in the right positions.

4. Soldering Quality:

Reheat any suspicious solder joints to ensure proper bonding and conductivity.
Remove excess solder or solder bridges with a soldering iron and desoldering wick.

5. Continuity Test:

Use a multimeter's continuity mode to verify that there are no open circuits or breaks in the traces.
Test the continuity of connections between the components and their corresponding pins on the PCB.

6. Signal Integrity:

Check if the GPIO pins on the Raspberry Pi are correctly connected to the LED and ADXL345 sensor.
Use a multimeter or oscilloscope to verify that the signals are reaching the correct pins.

7. I2C Communication:

Verify the I2C communication between the Raspberry Pi and the ADXL345 sensor.
Check the sender's address and the Raspberry Pi's I2C settings in the software code.

8. LED Behavior:

If the LED is not blinking as expected, review the software code controlling the LED.
Confirm that the GPIO pin settings and blinking pattern are correctly programmed.

9. Sensor Data:

If the LED behaviour is influenced by the ADXL345 sensor's data, ensure that the sensor is correctly connected and functioning.
Check if the sensor is providing accurate acceleration data to the Raspberry Pi.

10. Component Replacement:

If a component is suspected to be faulty, consider replacing it with a new one and retest the circuit.
Ensure that the replacement component is of the same type and specifications.

11. Isolation Testing:

Isolate sections of the circuit to identify specific problem areas.
Test individual components or sections of the PCB to pinpoint the source of the issue.

12. Consult Documentation:

Refer to datasheets and technical documentation for components to ensure correct usage and connections.

13. Seek External Help:

If the troubleshooting steps prove challenging, seek advice from experienced individuals, online communities, or forums.

14. Record Changes and Repeat Testing:

Document any changes made during troubleshooting, including component replacements, adjustments to code, or modifications to connections. After making changes, repeat testing to verify whether the issue has been resolved.

Major Lessons learned from Troubleshooting

1) Thoroughness: Troubleshooting requires a systematic and comprehensive approach. Skipping steps or making assumptions can lead to overlooking critical issues.

2) Visual Inspection: A visual inspection is often the first step and can reveal soldering errors, misplaced components, or visible damage that may cause circuit malfunctions.

3) Documentation: Keeping a record of changes, observations, and steps taken during troubleshooting helps in tracking progress, avoiding repetition, and sharing insights with others.

4) Setting Checkpoints: Checking power, ground, continuity, and signal integrity at various stages of troubleshooting ensures a holistic assessment of the circuit's health.

5) Component Interaction: Troubleshooting involves understanding how components interact within the circuit. Identifying relationships between different elements can help isolate problems.

6) Data Verification: If the circuit's behaviour is influenced by data from external sources (like the ADXL345 sensor), verifying the accuracy and integrity of that data is crucial.

Bibliography

References:

Parents. (n.d.). Does Your Baby Need a SIDS Monitor? Parents. Retrieved from https://www.parents.com/baby/health/sids/does-your-baby-need-a-sids-monitor/

Sensor Datasheet:

Analog Devices, Inc. (n.d.). ADXL345 Datasheet. Retrieved from https://learn.adafruit.com/adxl345-digital-accelerometer/pinouts

Raspberry Pi Documentation:

Raspberry Pi Foundation. (n.d.). Raspberry Pi Documentation. Retrieved from https://www.raspberrypi.org/documentation/

Python Programming Guide:

Python Software Foundation. (n.d.). Python Documentation. Retrieved from https://docs.python.org/3/

GitHub Repository:

BioBytes GitHub Repository.
https://github.com/ZoyebaMahbub5837/InfantHealthMonitor

Adafruit ADXL34x Library:

Adafruit Industries. (n.d.). Adafruit ADXL34x Python Library. Retrieved from https://github.com/adafruit/Adafruit_CircuitPython_ADXL34x

Electronic Component Suppliers:

Digi-Key Electronics. (n.d.). Electronic Components Distributor - Digi-Key Electronics. Retrieved from https://www.digikey.com/
SparkFun Electronics. (n.d.). SparkFun Electronics. Retrieved from https://www.sparkfun.com/

Technical Forums and Communities:

Raspberry Pi Forums. (n.d.). Retrieved from https://www.raspberrypi.org/forums/
Stack Overflow. (n.d.). Stack Overflow - Where Developers Learn, Share, & Build Careers. Retrieved from https://stackoverflow.com/

Appendix

Hardware Python Code: -

```
import time
import board
import busio
import adafruit_adxl34x
import RPi.GPIO as GPIO
import threading

# setup adxl i2c functions
i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)

# Set up the LED and GPIO pins
led_pin = 21  # GPIO pin number used for the LED circuit
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

def adxl_senses():
    while True:
        print("%f %f %f"%accelerometer.acceleration)
        time.sleep(1)


def blink_led():
    while True:
        GPIO.output(led_pin, GPIO.HIGH)
        print("LED ON")
        time.sleep(0.5)
        GPIO.output(led_pin, GPIO.LOW)
        print("LED OFF")
        time.sleep(0.5)

try:
    # Create and start the LED blinking thread
    led_thread = threading.Thread(target=blink_led)
    led_thread.start()

    # Create and start the accelerometer thread
    adxl_thread = threading.Thread(target=adxl_senses)
    adxl_thread.start()

    # Wait for both threads to finish (which will not happen in this case)
    led_thread.join()
    adxl_thread.join()

except KeyboardInterrupt:
    # Clean up GPIO on keyboard interrupt
    GPIO.cleanup()
```