# Table of Contents

## Overview
- Parts

## Build
- Materials
- Build

## Code
- Prerequisite Internet Accounts
- Software
- Setting up IFTTT
- Installing the monitor

## Training an Anomaly Detector
- Python Virtual Environment Script
- Train Your Model
- About Autoencoders
- Further Reading

## AI Freezer Monitor
- Code
- Flashing your board
- Installing
Overview

The AI Freezer Monitor is a DIY IoT monitor that uses machine learning (ML) to provide early warnings of potential equipment failure. This guide covers building the device, collecting training data, setting up email alerts, training a custom auto encoder machine learning model, and deploying the model to a Feather HUZZAH32 development board.

The project is designed to be functional for low temperature scientific freezers (-60 C), with the goal of reducing catastrophic failure and the need to keep backup freezers running full-time. However, please note that this project is primarily for demonstrative and educational purposes and has not gone through extensive testing.

This project takes about two to three hours to fully complete. But the device will need to passively collect temperature data for about 30 days before you will be able to train the machine learning model.
Parts

Adafruit HUZZAH32 – ESP32 Feather Board (pre-soldered)
Aww yeah, it's the Feather you have been waiting for, this time with pre-assembled headers! The HUZZAH32 is our ESP32-based Feather, made with the...
https://www.adafruit.com/product/3591

Adafruit MCP9600 I2C Thermocouple Amplifier
Thermocouples are very sensitive, requiring a good amplifier with a cold-compensation reference. The Adafruit MCP9600 does all that for you, and can be easily...
https://www.adafruit.com/product/4101

Thermocouple Type-K Glass Braid Insulated
Thermocouples are best used for measuring temperatures that can go above 100 °C. This is a bare wires bead-probe which can measure air or surface temperatures. Most inexpensive...
https://www.adafruit.com/product/270
Build

This part of the tutorial covers everything you'll need to build and run your own freezer temperature monitor and alert system. We'll cover how to build the device, setting up a free online database, flashing code onto the device, monitoring the database and sending email alerts. The components used in this project can measure temperatures as low as -200 Celsius.

This part of the project takes about 60 to 90 minutes to build.
Materials

a) Adafruit Feather Huzzah32

b) Adafruit MCP9600

c) Adafruit Type K Thermocouple

d) Jumper Wires

e) Breadboard

Build

1. Solder the screw terminal to the top side of your MCP9600.
2. Solder the pins to the bottom side of the MCP9600.

Tip: Put the in the breadboard to hold them in place when soldering.

3. Insert the ESP32 and thermocouple amplifier into the breadboard.

4. Follow the wiring diagram below to connect the thermocouple amplifier to the ESP32 using your jumper wires.
And a view when complete:

### Code

#### Prerequisite Internet Accounts

1. An Adafruit IO account, you can sign up for free [here](#).
2. An IFTTT Account, you can join IFTTT [here](#).

#### Software

Clone this [repo](#) to your computer, or you can download the directory for this project [here](#).

Once you downloaded the project files you'll need to create a config.h file and fill in your Adafruit IO and WiFi credentials.

1. In the FreezerTempAlert folder create a new text file called config.h
2. Copy and paste the code below into that file and save:

```c
/************************ Adafruit IO Config *******************************/
// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME "your-username"
#define IO_KEY "your-key"

/******************************* WIFI **************************************/
#define WIFI_SSID "your-ssid"
#define WIFI_PASS "your-password"
```

©Adafruit Industries
#include "AdafruitIO_WiFi.h"

AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

3. Find your Adafruit IO credentials in the top right corner of the Adafruit IO webpage.

4. Copy and paste them in here between the quotes:

```cpp
#define IO_USERNAME "your_username"
#define IO_KEY "your_key"
```

Remember to treat your key like a password

5. Next you will need to add the WiFi SSID, or network name, and the password. Note: The code doesn't support WiFi networks that use a captive portal.

```cpp
#define WIFI_SSID "your_ssid"
#define WIFI_PASS "your_pass"
```

6. Open FreezerTempAlert.ino and change the threshold temperature to the temperature you wanted to be alerted at. The default temperature is -50 Celsius.

```cpp
// Set the temperature you want to be alerted at
const float thresholdTemperature = -50;
```

Flashing the code

1. Follow the first section of this guide to add the ESP32 extension to the Arduino IDE.
2. Install the libraries listed below using Arduino's Library manager. If you have used the library manager before, here is a helpful guide.
   - Adafruit IO Arduino
Adafruit MCP9600

- Note: If you're prompted to install other dependencies for these libraries, select Install All

3. With FreezerTempAlert.ino open in the Arduino IDE select Tools > Board > Adafruit ESP32 Feather
4. Select the active COM Port in Tools > Port
5. Press the Upload button

6. Open the Arduino serial monitor to verify that everything is working properly on the board.

7. Go back to Adafruit.io and check that both the temperature feed and the temp_alert feed were created.

Setting up IFTTT

You'll use If This Then That (IFTTT) to set up automated alerts to your email any time the temperatures goes above a threshold. The threshold is set to -50c by default.

1. Log in to your IFTTT account and press the Create button in the top navigation.
2. Select the + button to add a trigger.

3. Search for Adafruit
4. Follow the prompts to connect and authenticate your Adafruit IO account.

5. Once your account is connected choose the Any new data trigger.

6. Select temp_alert from the drop down menu and press Create trigger

7. Select the next + to set your action.
8. Scroll down or search for the Email action

If this is the first time you're using the email action a pin will be sent to you to confirm that you are the owner of the email account.

9. Customized the email message and select create action

10. Finally select finish, your IFTTT service is ready to go.
Installing the monitor

Now that you project is built and your services are setup you can install the thermocouple in the freezer. You can use a binder clip or a magnet to keep it in place. The Feather Huzzah32 board should be kept outside the freezer. Many breadboards come with an adhesive back, you can use that to secure your project to the side of the freezer.

Training an Anomaly Detector

Once you have 30 days of data from your freezer, you're ready to train a custom Machine Learning (ML) model! The goal of this model is to warn you before a freezer fails, giving you time to address the failure prior to losing critical work. To do this we'll use an auto-encoder algorithm.

Environment Setup

We'll use Python to train the model. If you're new to Python, check out this guide for a quick introduction.

We'll also use several machine learning Python libraries. We recommend using these in a virtual environment as this will keep your global installation of Python clean and keep all the dependencies for this project in one place.
Download the autoencoder_training.ipynb and either the win_python_venv.bat file for Windows or the mac_linux_python_venv.sh for Mac or Linux files from this repo. You may already have them if you cloned the repo in the previous step.

Python installation

You'll also need to install Python. Follow the links below to install Python for your appropriate operating system.

Python 3.8.6 Windows Installation

Python 3.8.6 Mac Installation

VS Code installation

You can use VS Code to view and run the Jupyter notebook that contains the anomaly detection training code. The notebook uses interactive Python, which walks you through the steps of the training the model. This guide will walk you through how to navigate Jupyter notebooks in VS Code. Use this link to download and install VS Code. After you install VS Code you will also need to install the Python extension.

Python Virtual Environment Script

First open the Jupyter notebook in VS Code. Then open the terminal by pressing `ctrl + shift + ` in the new terminal window type win_python_venv.bat or mac_linux_python_venv.sh and press enter.
Train Your Model

The rest of the tutorial is in the Jupyter notebook, make sure the new virtual environment you created is activated.

About Autoencoders

An autoencoder is a type of compression algorithm that works well on the type of data it's trained on, but not so well on any other data. For anomaly detection, you can use this characteristic of the algorithm to check whether new data is similar to or different than the training data used.

The model is built using two parts: 1) an encoder, which compresses the data, and 2) a decoder, which decompresses the data. Decompression allows you to check the accuracy of the re-created data. A high error rate in re-created data is likely an anomaly.

Further Reading

- Autoencoder example from Keras
- Autoencoder example from TensorFlow
AI Freezer Monitor

With your model trained and exported to a C array you're ready to implement it on your Feather Huzzah 32. We'll need to update the software on our ESP32, but the hardware can stay the same. In fact, it is best to leave the thermocouple in the freezer since moving it may affect the results of the machine learning model.

()Code

Clone this repo to your computer, or you can download the directory for this project here.

Once you have downloaded the project files, you'll need to create a config.h file and fill in your Adafruit IO and WiFi credentials.

1. In the AlFreezerMonitor folder create a new text file called config.h

   Note: You can also copy the config.h file you created for the data collection code.

2. Copy and paste the code below into that that file and save:

   ```
   /*******************************************************************************
   Adafruit IO Config
   ******************************************************************************/
   // visit io.adafruit.com if you need to create an account,
   // or if you need your Adafruit IO key.
   #define IO_USERNAME "your-username"
   #define IO_KEY "your-key"
   /*******************************************************************************
   WIFI
   ******************************************************************************/
   #define WIFI_SSID "your-ssid"
   #define WIFI_PASS "your-password"
   #include "AdafruitIO_WiFi.h"

   AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
   ```

3. Find your Adafruit IO credentials in the top right corner of the Adafruit IO webpage.
4. Copy and paste them in here between the quotes:

```c
#define IO_USERNAME "your_username"
#define IO_KEY "your_key"
```

Remember to treat your key like a password

5. Next you will need to add the WiFi SSID, or network name, and the password. Note: The code doesn't support WiFi networks that use a captive portal.

```c
#define WIFI_SSID "your_ssid"
#define WIFI_PASS "your_pass"
```

6. Move the model_data.h file you created earlier into the AlFreezerMonitor directory

**Flashing your board**

1. Follow the first section of this [guide](#) to add the ESP32 extension to the Arduino IDE.

2. Install the libraries listed below using Arduino Library Manager. If you haven't used the library manager before, [here](#) is a helpful guide.

   1. Adafruit IO Arduino
   2. Adafruit MCP9600
   3. TensorFlowLite_ESP32

   Note: If you're prompted to install other dependencies for these libraries, select Install All
3. With FreezerTempAlert.ino open in the Arduino IDE select Tools > Board > ESP32 Arduino > Adafruit ESP32 Feather

4. Select the active COM Port in Tools > Port

5. Press the Upload button
Installing

If you moved the thermocouple to upload the new code, do your best to re-install it in the same place. If the thermocouple is moved too far away from the training position, the ML model may mistake normal operation as an anomaly.

Your IFTTT warning should still be working the same as before, so you don’t need to change anything there.