Purpose of the system

Student Guidance:

- Explain **what is the purpose** of your system (the overall goal) ,mention what **environmental issue** it addresses.

  Note: Keep this to **one short sentence** that introduces the rest of your design.

Sample: Purpose

The **purpose** of this system is to **monitor** key **environmental indicators** of **forest stress over time**, such as temperature, light, and soil moisture, in order **to support the modelling** of **early** drought or wildfire risk.

Note: Yours purpose must be more specific.

Design Objectives:

Student Guidance:

Design Objectives: A **list** of **things** your **system must** be **able to do**, one by one. Each **objective** should describe a specific action your system will perform.

The system **will** use environmental sensors such as temperature and soil moisture as inputs and provide at least one output on the Micro:bit, operating automatically once started. {BR1}
The system **will** collect and store in a .csv file, environmental data related to forest conditions (for example, temperature or moisture readings) for later analysis. {BR2}
The system **will** simulate a forest-related process or risk over time, such as drought or wildfire risk, by changing a risk level as conditions persist, and will be tested using different input values to show how it responds. {BR3}

The system will use Python to model a forest-related disaster risk (e.g. drought or wildfire risk) using data collected from the embedded system (e.g. soil moisture and temperature), combined with simulated data (e.g. generating rainfall values over time) or open-source data (e.g. Met Éireann weather data or published climate datasets) if required. {AR1}

The system will test how the model behaves under different conditions by changing one or more key variables (e.g. increasing temperature, decreasing rainfall, or lowering soil moisture) and observing how the predicted risk level changes. {AR2}

The system or model will automatically adapt to changing conditions using feedback, such as triggering a high-risk LED warning pattern, warning icon, or buzzer alert when risk becomes high, or automatically updating the risk prediction using the most recent sensor readings as conditions change. {AR3}

Sample: Design Objectives

The **system will use environmental sensors**, such as a temperature sensor and a soil-moisture sensor, as inputs and will provide an output on the Micro:bit in the form of warning icons and risk-level indicators. Once the system is started, it will operate automatically without further user input. The system will **collect and store** environmental data, such as temperature and soil-moisture readings, in a CSV file so that the data can be examined later. The **system will simulate** a forest-related process over time by increasing or decreasing a drought-risk level as conditions persist or change, and it will be tested using different input values to show how the system responds under different environmental conditions.

The system **will** use Python to **model a forest-related disaster risk**, such as drought risk, using soil moisture and temperature data collected from the embedded system. This data will be combined with simulated data, such as generated rainfall values over time, to allow the model to represent changing environmental conditions. The system **will** test how the **model behaves under different conditions** by changing key variables, such as increasing temperature or decreasing rainfall, and observing how the predicted drought risk level changes in each scenario. The system **will automatically adapt** to changing conditions using feedback, for example by triggering a high-risk warning icon or LED pattern on the Micro:bit when drought risk becomes high, and by updating risk predictions based on the most recent sensor data.

Guidance and Sample: Project Options

For my chosen project, a **forest drought-monitoring system**, I considered **two different ways to build the SAME system** for collecting and processing environmental data.

Option 1: Continuous Data Logging

In this option, the Micro:bit would record soil moisture and temperature readings at regular time intervals and send all data to Python for storage and analysis. This approach would capture every small change over time and produce a large dataset. However, it would also generate unnecessary data, include minor fluctuations that are not meaningful, and make the drought-risk modelling more complex and harder to interpret.

Option 2: Threshold-Based Data Logging
In this option, the Micro:bit would only record and store data when soil moisture or temperature moves **outside a normal range**. The Micro:bit would give an immediate warning when dry conditions are detected, and Python would model increasing drought risk based on repeated threshold breaches over time. This approach reduces noise in the data and focuses on meaningful environmental changes.

Guidance and Sample: Justification of Design Choices

I chose **Option 2** because it aligns more closely with my **design objectives**. It focuses on significant changes in forest conditions rather than minor fluctuations, keeps the data manageable, and supports clearer **rule-based simulation for BR3** and **pattern-recognition modelling for AR3**. As a result, the system is easier to understand, more reliable, and better suited to demonstrating environmental change over time.

Stakeholders and End Users

Student Guidance

Stakeholders: Stakeholders are people or groups who **benefit from or are affected by the system**, even if they do not use it directly.

environmental or **conservation** organisations interested in forest risk/stress {name}
teachers or schools interested in modelling environmental change
community groups concerned about drought or wildfire awareness
organisations promoting **climate** or **STEM** education

For stakeholders, briefly state:

what they **need**
why they **need** it
how your system helps **meet** that **need**

End Users: End users are the people who **directly use or operate the system**.

a **student** collecting data and running the Python model
a **teacher** demonstrating the system in a classroom
a **learner** testing "what-if" scenarios

For end users, explain:

what they need from the system
how your system supports them through clear **feedback** and/or **modelling**

Sample Answer

The main **stakeholders** for this system include environmental organisations, teachers, and community groups {list your own} who want clear ways to understand and explain how forest conditions change over time. They need tools that can show how risks such as drought or wildfire develop, and this system helps by modelling rising risk levels using real environmental data in an easy-to-understand way.

The **end user** is a student or learner who directly operates the system to collect data and explore the model. They need clear feedback and simple visual warnings that show how environmental risk increases or decreases. The system supports this by detecting changes in temperature and soil moisture, simulating rising risk over time, and displaying clear warning indicators on the Micro:bit.

Sample: Technologies That Will Be Used

Embedded System Technologies

Micro:bit: Used to collect environmental data and provide immediate feedback.

Sensors (e.g. temperature, soil moisture, light): Used to measure environmental conditions related to forest stress.

Outputs (LEDs, icons, buzzer): Used to display warnings or risk levels when conditions change.

Software Technologies

MakeCode / MicroPython: Used to program the Micro:bit to read sensors and control outputs.

Python: Used to analyse the collected data, model environmental risk, and run what-if simulations.

Data Collection and Storage

Environmental readings will be logged and stored in a **CSV file**, allowing the data to be analysed later by the Python model.

Modelling Technologies

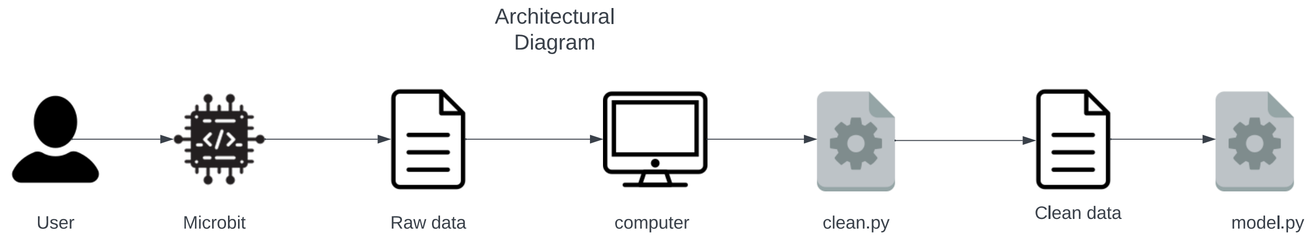
A **simple rule-based model** will be used in Python to simulate changes in drought or wildfire risk based on sensor data over time.

System Architecture

Student Guidance:

System Architecture: In both the Embedded System and Python Model, **data** is moving from one part of the system to another. A **flowchart** is the diagram that shows this movement clearly by illustrating the **direction** and **flow** of data **through the main components**.

Overall System Architecture: Overall System Architecture Diagram.



Flow Chart 1 {Embedded System}

Your embedded system flowchart should show **how data moves through the system**, from the environment to the output, and then repeats automatically.

Use **one box per step**, in this order:

1. **Environment** What is being measured e.g. soil moisture, temperature, light level
2. **Sensors (Inputs)** What sensors used to collect data e.g. soil moisture, temperature, light sensor
3. **Micro:bit Reads Data Values:** from the sensors, e.g. analogue or digital sensor readings
4. **Apply Rules / Simulate Process** **if** moisture low → increase drought risk
   if temp high → increase fire-risk
   if light low → canopy simulation
   if heart rate high → alert
5. **Outputs** How the system gives feedback e.g., LED symbol - Neopixels - buzzer alert - servo movement - display value.

Data Logging To computer, e.g., send data over serial - validate readings - save to CSV

Repeat Loop e.g., pause or go back to start

Explanation: This flowchart shows how my embedded system collects environmental data, applies simple rules to simulate a forest-related process, provides feedback using outputs, logs data if required, and then repeats the cycle automatically.

Flow Chart 2 {Python Model}

Import Data e.g., read CSV file - load dataset - open logged readings

Clean & Validate Data e.g., remove impossible values - handle missing readings - convert data types

Analyse Data e.g., calculate averages - find patterns - identify trends - compare variables

Apply Model e.g., rules-based model - threshold model - simple probability - decision-tree style logic

Generate Output / Results e.g., risk score - prediction - classification - "what-if" comparison

What-If Scenarios (If Used) e.g., increase temperature - reduce rainfall - change canopy cover - increase density

User Interpretation e.g., student interprets results - teacher reviews model - simple recommendations

Explanation: This flowchart shows how my Python model loads the CSV data, cleans and analyses it, applies a simple model to calculate risk or patterns, and then produces outputs or what-if results that the user can interpret

Use one box per step, in this order:

1. **Input Data:** What data the model uses
   e.g. CSV file from Micro:bit, simulated data, or open-source data

2. **Load Data into Python** How the data is brought into the program e.g. read CSV file, load dataset

3. **Clean and Validate Data** Make sure the data is usable
   e.g. remove impossible values, handle missing readings, convert data types

4. **Analyse Data** Look for patterns or trends
   e.g. calculate averages, check changes over time, compare variables

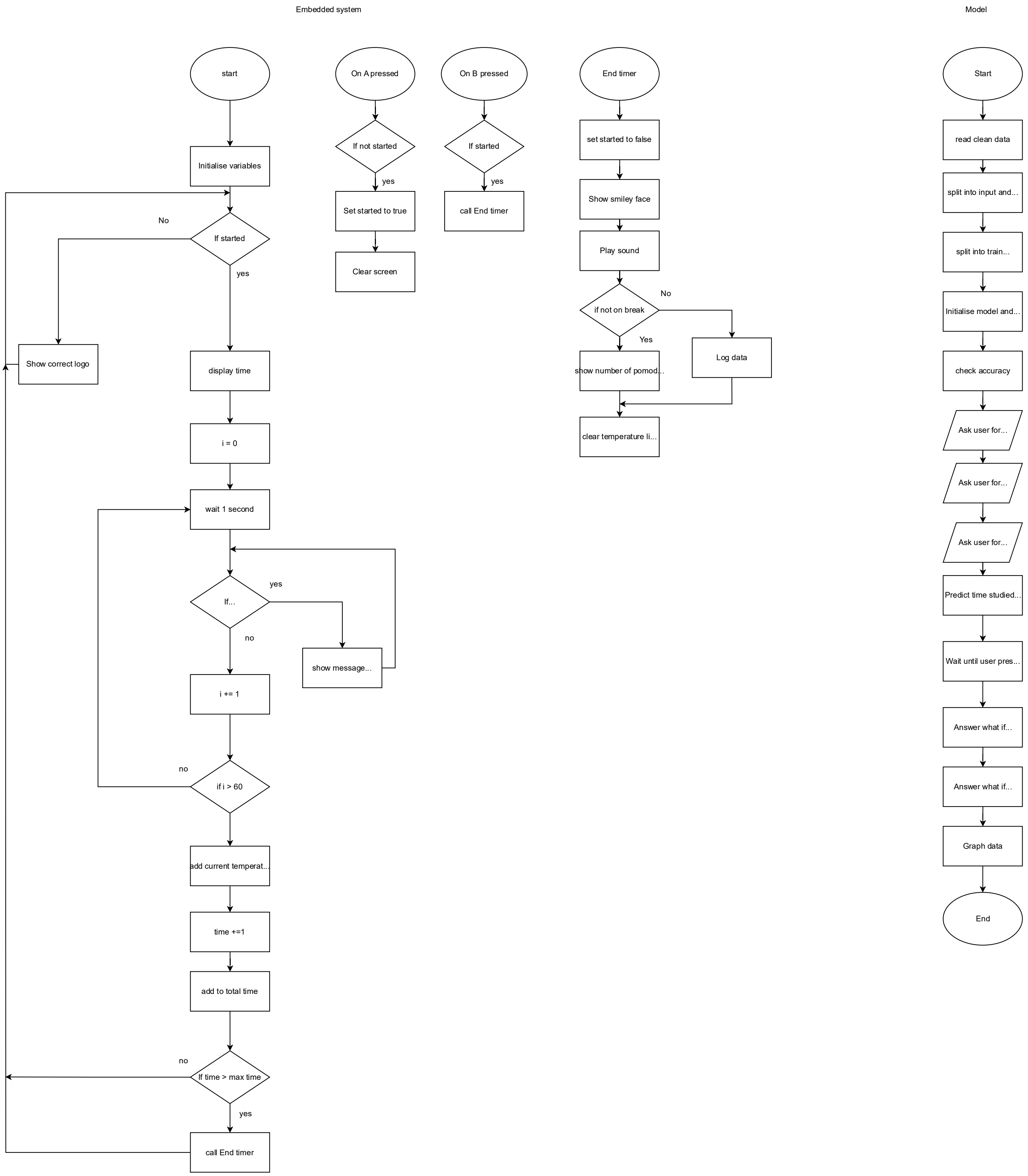
5. **Apply Model / Rules** Use a simple model to calculate risk
   e.g. if average moisture is low → increase drought risk
   if temperature stays high → increase wildfire risk

6. **Generate Output / Results** What the model produces
   e.g. risk score, risk level, prediction, classification

7. **What-If Scenarios (If Used)** Test different conditions e.g. increase temperature, reduce rainfall, change moisture levels

8. **User Interpretation** How results are understood e.g. student interprets risk level, teacher reviews results

Explanation: This flowchart shows how my Python model loads logged environmental data, cleans and analyses it, applies a simple rules-based model to calculate forest-related risk, runs what-if scenarios where required, and produces outputs that the user can interpret.


Embedded system
Model
start
Initialise variables
No
If started
yes
Show correct logo
display time
i = 0
wait 1 second
yes
If...
no
show message...
i += 1
no
if i > 60
add current temperat...
time +=1
add to total time
no
If time > max time
yes
call End timer
On A pressed
If not started
yes
Set started to true
Clear screen
On B pressed
If started
yes
call End timer
End timer
set started to false
Show smiley face
Play sound
No
if not on break
Yes
show number of pomod...
Log data
clear temperature li...
Start
read clean data
split into input and...
split into train...
Initialise model and...
check accuracy
Ask user for...
Ask user for...
Ask user for...
Predict time studied...
Wait until user pres...
Answer what if...
Answer what if...
Graph data
End