# BGU IIOT DDS Assignment

## System Overview

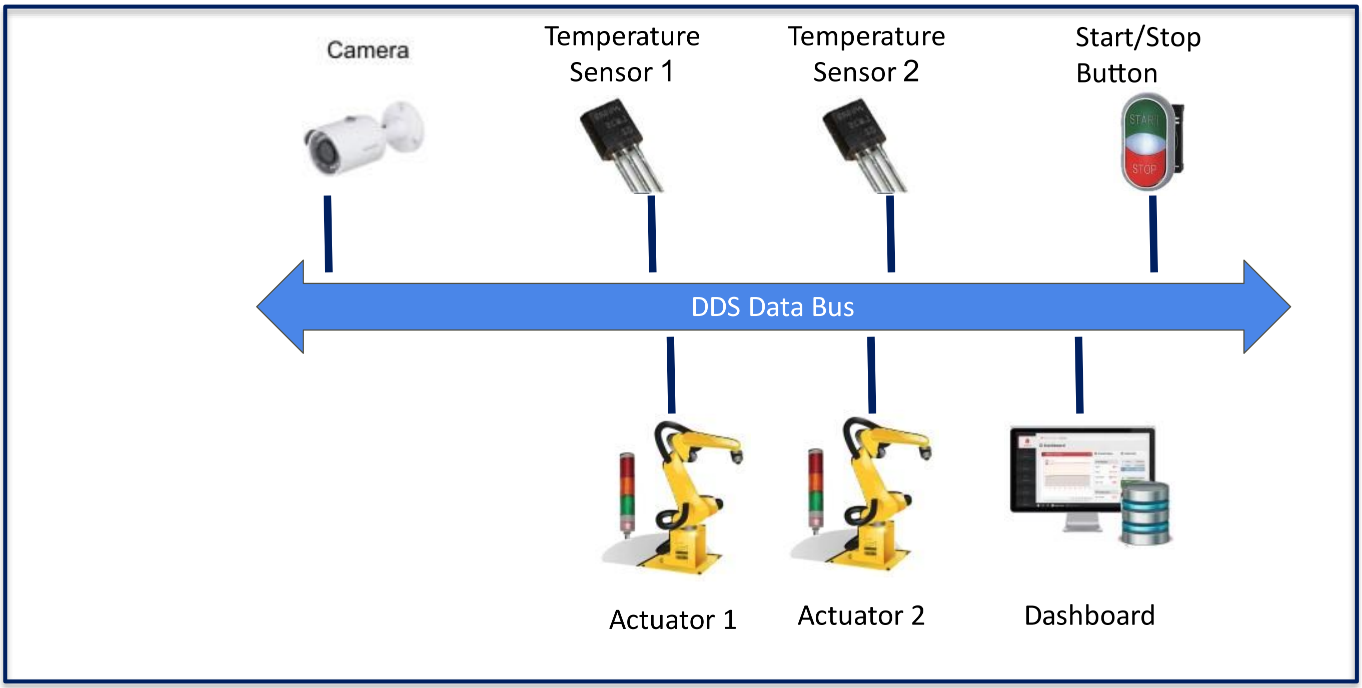


Figure 1: Connectivity Architecture

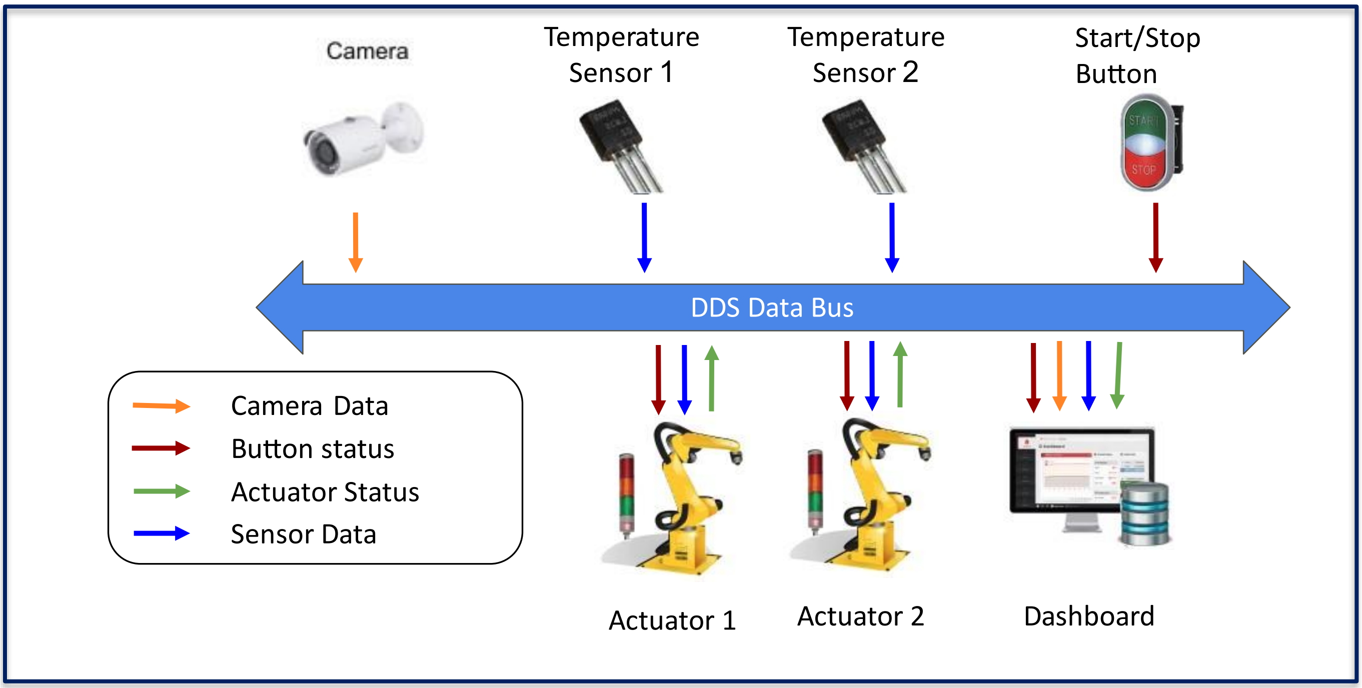


Figure 2: Connectivity Architecture – Data Flow View

# Application’s Logic Description

## Camera (5)

- Sends the current state at 10 Hz (every 0.1 seconds).
- For the sake of simplicity, the message will be a string with the real time in it.
- It is only important to look at the current state, **there is no need to ensure the arrival of each message and no need for the previous messages here**.
- The Camera shall also print the message to a console.

## Start/Stop Button (10)

- Sends a Starts/Stops command (default is Start).
- For the sake of simplicity:
  - A stop command shall be sent every 20 seconds.
  - 5 seconds after the stop command, a start command shall be sent.
- The Button shall also print the command to a console.

## Temperature sensor 1  (10)

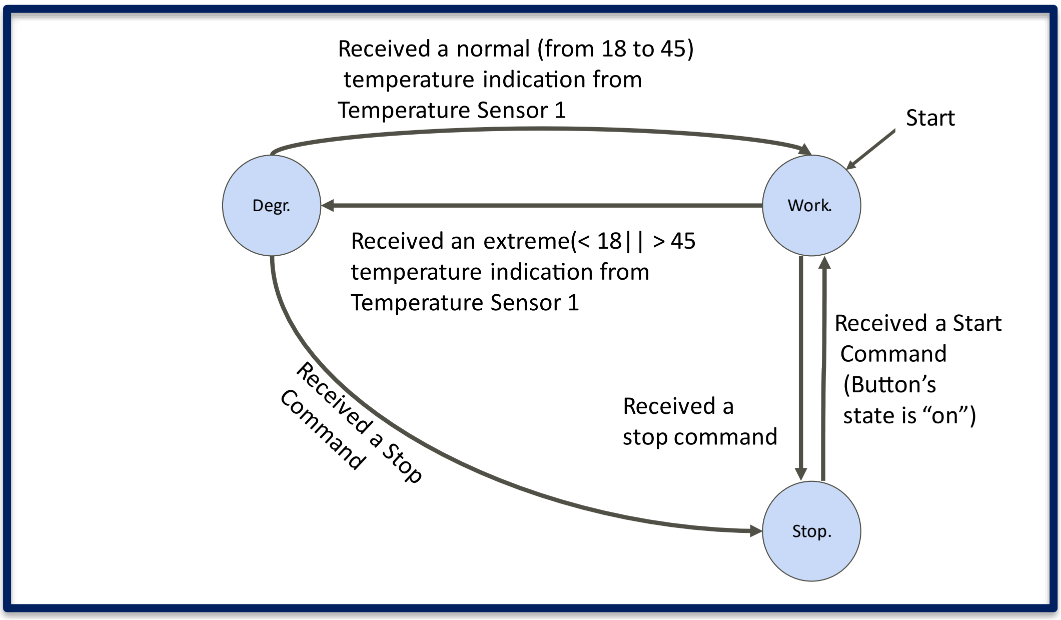
- The sensors shall publish a temperature reading at 1 Hz (every 1 seconds).
- For the sake of simplicity, the temperature shall be a random value 10 - 60 degrees (integer).
- The sensor shall also print the temperature to a console.
- While the actuator is in stopping status, the sensors shall not print.

## Temperature sensor 2  (10)

- The sensors shall publish a temperature reading at 10 Hz (every 0.1 seconds).
- For the sake of simplicity, the temperature shall be a random value 0 - 50 degrees (integer).
- The sensor shall also print the temperature to a console.
- While the actuator is in stopping status, the sensors shall not print.

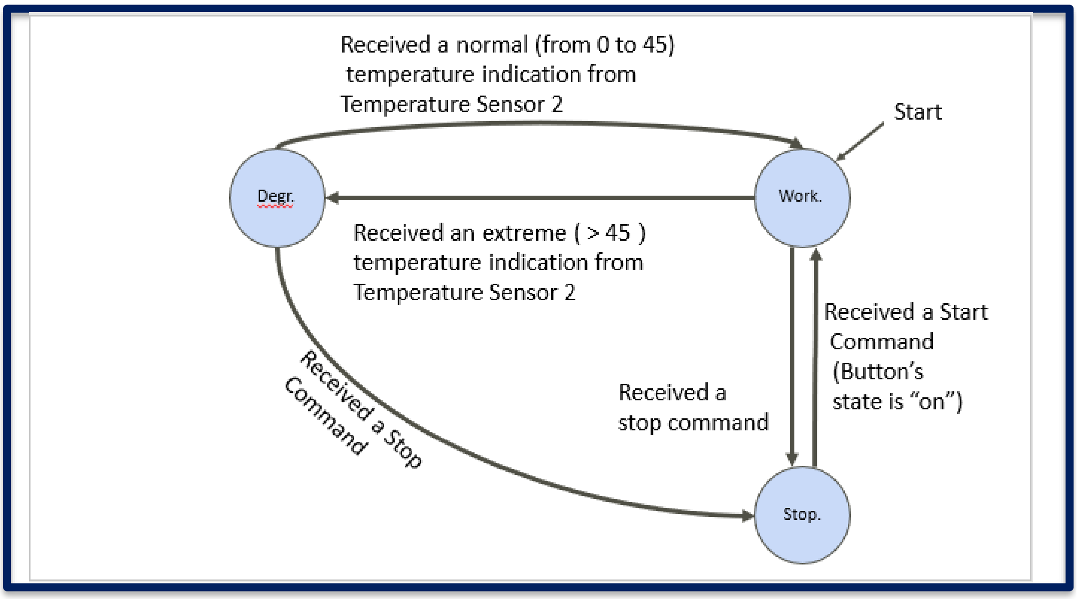
# Actuator 1 (15)

- The actuator shall publish its current status **upon change** [Working, degraded, Stopped].
- The actuator shall print its status to a console at 1 Hz (every second).
- The actuator shall change its status according to the following state machine:


Received a normal (from 18 to 45) temperature indication from Temperature Sensor 1
Start
Degr.
Work.
Received an extreme(< 18|| > 45 temperature indication from Temperature Sensor 1
Received a Start Command (Button's state is "on")
Received a Stop Command
Received a stop command
Stop.

## Actuator 2 (15)

- The actuator shall publish its current status **upon change** [Working, degraded, Stopped].
- The actuator shall print its status to a console at 1 Hz (every second).
- The actuator shall change its status according to the following state machine:


Received a normal (from 0 to 45) temperature indication from Temperature Sensor 2
Start
Degr.
Work.
Received an extreme ( > 45 ) temperature indication from Temperature Sensor 2
Received a Start Command (Button's state is "on")
Received a Stop Command
Received a stop command
Stop.

## Dashboard (20)

- The Dashboard shall print the status of the system to a console every 5 seconds.
- The Dashboard shall display the last 10 measurements **only** of an extreme temperature from Temperature sensor 1 and from Temperature sensor 2 (**even if the dashboard was initialized after the sensors sent their last status messages**.
- The Dashboard shall display the last 10 statuses of the actuator **even if the dashboard was initialized after the actuators sent their last status messages**.
- The template of the dashboard prints to a console:

    - `Camera: <the` *`last message received from the Camera>`*
    - `Actuator 1: [`*`list of last 10 published statuses`*`]`
    - `Actuator 2: [`*`list of last 10 published statuses`*`]`
    - `Thermometer 1: [`*`list of last 10 extreme temperatures`*`]`
    - `Thermometer 2: [`*`list of last 10 extreme temperatures`*`]`

# What to do?

- Implement the application logic described above and use DDS capabilities as you see it right. Our recommendations:
    - Understand fully the assignment and what needs to be done
    - Understand which types of messages, topics, participants and QoS will be used in the system
    - Make a small publisher - subscriber example for yourself to play with different types of QoS
    - Implement each element one at a time and check the system (by running) after each step
    - Run the full system, see if everything works as expected
    - Submit
- Serve 7 python files (one for each component) + DDS XML configuration file (named DDS.xml) in a zip over Moodle. The name of the file shall include the IDs of both students.
- The Actuators' IDs should also correspond to the IDs of both students.

# The grade

| | |
|---|---|
| ❖ Camera | 5 |
| ❖ Start/Stop Button | 10 |
| ❖ Temperature sensor 1 | 10 |
| ❖ Temperature sensor 2 | 10 |
| ❖ Actuator 1 | 15 |
| ❖ Actuator 2 | 15 |
| ❖ Dashboard | 20 |
| ❖ Readability and clarity of the code | 15 |