

Automated Ambient System (AAS)

Ahmad Mady, Amro Hassan, Mostafa Kashaf, Marwan Sallam, Mohamad Magdy

December 26, 2022

1 Introduction

Warehouse automation is the process of automating the movement of inventory into, within, and out of warehouses to customers with minimal human assistance. As part of an automation project, a business can eliminate labor-intensive duties that involve repetitive physical work and manual data entry and analysis.

For example, a warehouse worker may load an autonomous mobile robot with heavy packages. The robot moves the inventory from one end of the warehouse to the shipping zone and software records the movement of that inventory, keeping all records current. These robots improve the efficiency, speed, reliability and accuracy of this task.

But warehouse automation does not require physical or robotic automation, and in many cases simply refers to the use of software to replace manual tasks. However, this scenario illustrates how robots and humans work together to accomplish repetitive tasks while minimizing fatigue and injury.

In an automated warehouse, not only machines and machine operations need to be automated, but also lights and cooling need to be also automated. In our project, an Automated Ambient System (AAS) is introduced that controls AC units and light systems. The system is classified as a dynamic hybrid concurrent system as there are time-driven and event-driven processes making it a hybrid system and a model for controlling the central condition unit is used to implement continuous real-time control. Concurrent states will be defined between the light and AC systems as well as between sensing and actuation for the systems. Emergency states are studied as in case of any fires or hazards in the warehouse. Light and Ac systems are controlled in a concurrent system.

2 Project Description

2.1 Inputs

There are a total of 6 inputs to the system, namely: Emergency Button, Power Button, Smoke Detector, Light Sensor, Temperature Sensor, and Humidity Sensor.

2.1.1 Emergency Button

The emergency button is a very important input, since it will be pressed in case of an emergency. It will toggle the circuit breaker and shut down most of the system to ensure that nothing catches fire or gets destroyed. The emergency button can then be de-pressed when the emergency is taken care of.

2.1.2 Power Button

The power button will turn the whole system on when pressed once. When it is pressed once again (de-pressed), it will turn off the system.

2.1.3 Smoke Detector

The smoke detector is a sensor that activates once it detects any type of smoke, where that input will toggle the circuit breaker to enable emergency mode.

2.1.4 Light Sensor

The light sensor is needed to detect the amount of natural light lighting up the warehouse. It will send the input to the controller to control the amount of lighting in the warehouse.

2.1.5 Temperature Sensor

The temperature sensor will detect the temperature of the warehouse and sense the input to the controller to control the temperature of the warehouse.

2.1.6 Humidity Sensor

The humidity sensor will detect the humidity of the warehouse and sense the input to the controller to control the humidity of the warehouse by controlling temperature.

2.2 Outputs

There are 4 outputs to this system namely: Circuit Breaker, Ventilation Fans, Light Bulbs, and AC Compressor and Fan. This system is a hybrid system because it has both continuous and discrete systems.

2.2.1 Circuit Breaker

The circuit breaker will toggle if either the emergency button is pressed, or the smoke detector detected a smoke. When the emergency is taken care of, the circuit breaker toggles back to its off state.

2.2.2 Ventilation Fans

The ventilation fans turn on in case there is an emergency to suck away any smoke that could have been detected. It turns off when the emergency is taken care of.

2.2.3 Light Bulbs

The light bulbs turn on when the power button is pressed. The bulbs' illumination percentage is changed based on the input received from the light sensor, which makes the lighting system a discrete event system.

2.2.4 AC Compressor and Fan

The AC Compressor turns on when the power button is pressed. The AC actuation will be continuously changing the temperature based on the inputs from the temperature sensor, which will be considered as the continuous time system controlled by the controller (Plant) block (like PID or Fuzzy) and the feedback loop. The fan will operate once the temperature sensor and the humidity sensor sense the required temperature and humidity, and the compressor will be automatically turned off to save electricity. The range for the temperature used is 0 Celsius to 45 Celsius.

2.3 Modes of Operation

The following will describe briefly what the states are and how the states will transition between each other in the Autonomous Ambient System (AAS).

It consists of two main superstates; Emergency and Normal. System will transition from normal state to emergency state if smoke is detected or emergency button is pressed. System will return back to normal state if emergency button is depressed and smoke is cleared. In the emergency state the following will happen: The lights will illuminate fully to increase visibility, ventilation fans will turn on to vent all the smoke out of the warehouse, the air conditioning will turn off, and the circuit breaker will turn on to prevent any electricity from causing another emergency. Under the normal state which is the default state there are two main superstates and they are ON and OFF. ON state is operating only if the Power Button pressed, while on the other hand, the OFF state is operating if the Power Button is depressed. The OFF State is the default substate, and everything does not work in this state. Next, under the ON superstate, there are two substates which operate concurrently. The substates are Lights and AC. Lights state includes two concurrent substates which are Light/Illumination Sensing and Light Actuation. Simply the ambient illumination readings from the light sensor will be used to control how intense the light of the bulbs should be to achieve the target ambient visibility, and this would save energy and prevent the surrounding from being too bright or too dim, for example, if it is daytime and there is enough light passing through to the warehouse, the need for high intense bulb lights will not be needed and will be redundant. States of light actuation will transition between each other according to needed light intensity to achieve target ambient visibility.

AC state includes three concurrent substates and they are Temperature Sensing and AC Actuation, and Humidity Sensing. Simply the ambient temperature readings from the temperature sensor will be used to control the AC temperature needed to achieve the target ambient temperature and this would again save energy, and will keep the products in the warehouse at optimum condition and also cool down the machines in the factory.

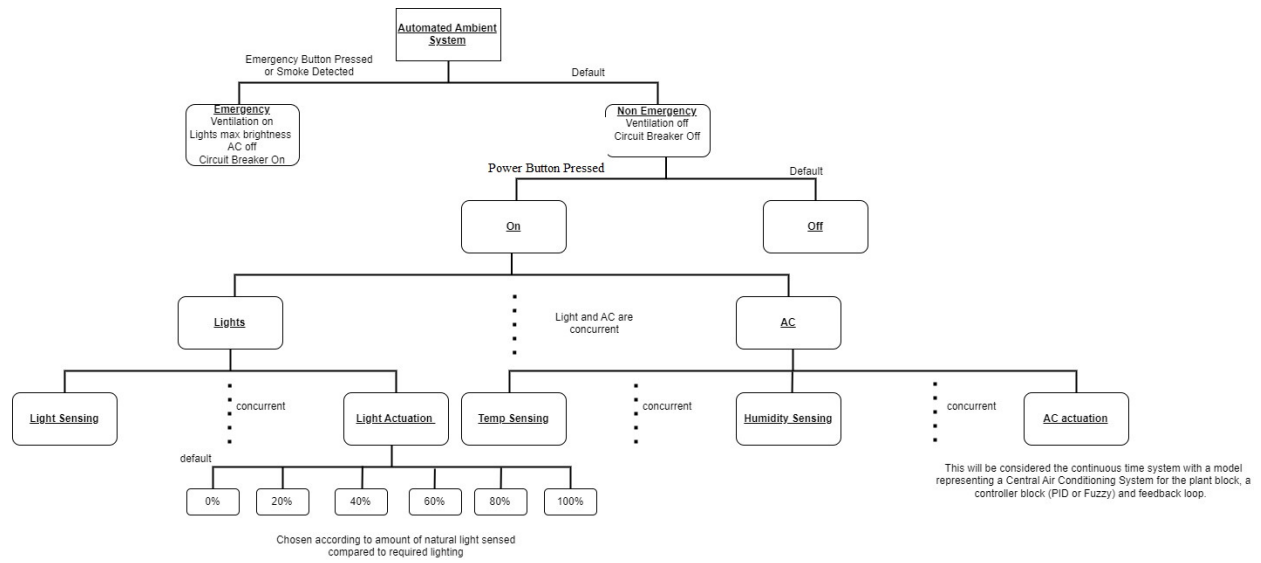


Figure 1: Flow Diagram of AAS

3 Project State Chart

3.1 State Chart Design

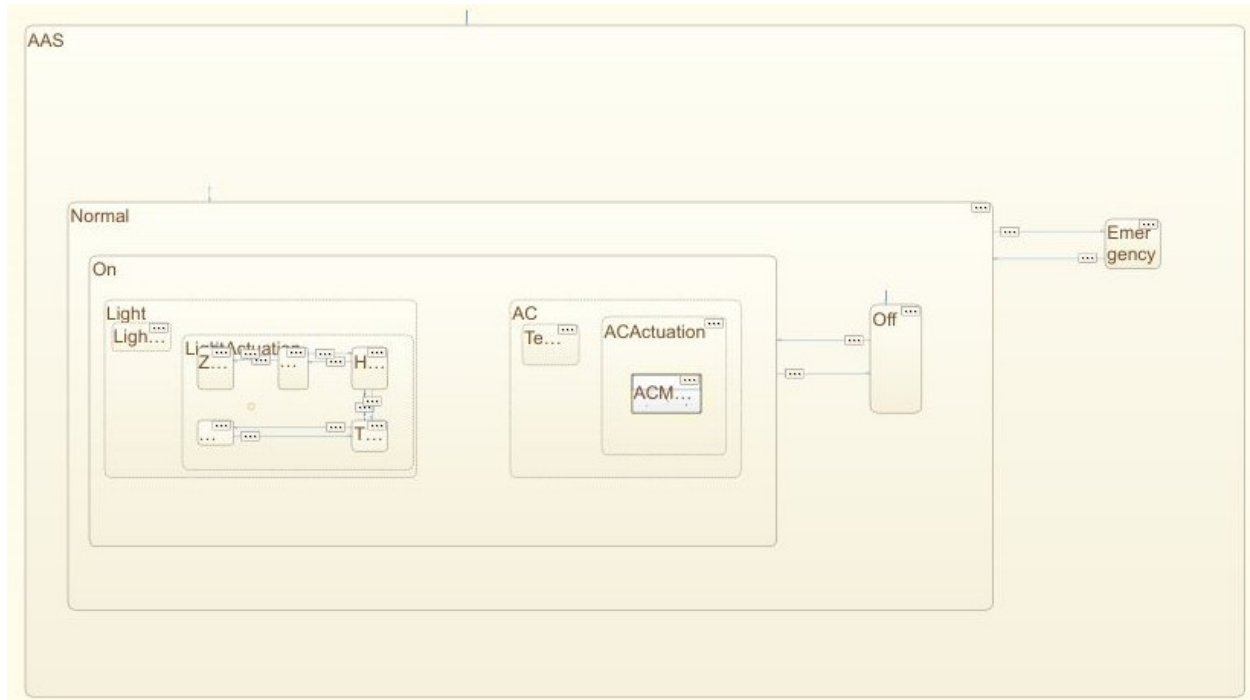


Figure 2: State Chart Design of AAS

In figure 2, we describe our system operation using state charts. Our system AAS includes the normal state (power) and the emergency state. In the normal state, we have the OFF State (default state, figure 6) and the ON State, which turns on the system itself (Light and AC) concurrently. In the Light State (figure 3, left), there is the LightSensing State which contains the light sensor that works concurrently with the Light Actuation State which is responsible for controlling the percentage of light illumination. In the AC State (figure 3, right), we have the TempSensing State which has the temperature sensor working concurrently with the ACActuation State, which is the continuous time state in our system that controls the motor speed of the AC based on the ambient temperature of the warehouse. Finally, the Emergency State (figure 5) enables when the Smoke Detector detects smoke or when the emergency button is pressed. Inside the ACActuation State, we implemented a Fuzzy Logic Controller (figure 12) to ensure that the AC reaches the desired temperature.

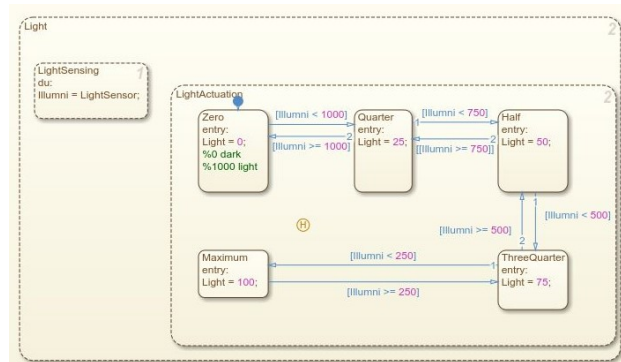


Figure 3: Light State and AC State

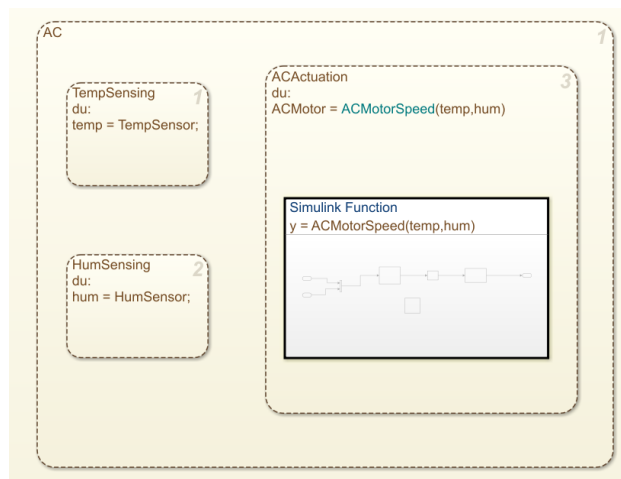


Figure 4: Temperature Sensing, Humidity Sensing, and the AC Sates

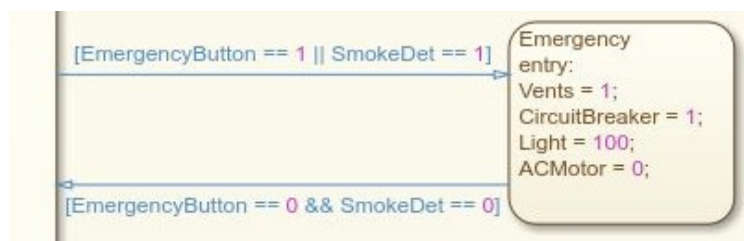


Figure 5: Emergency State

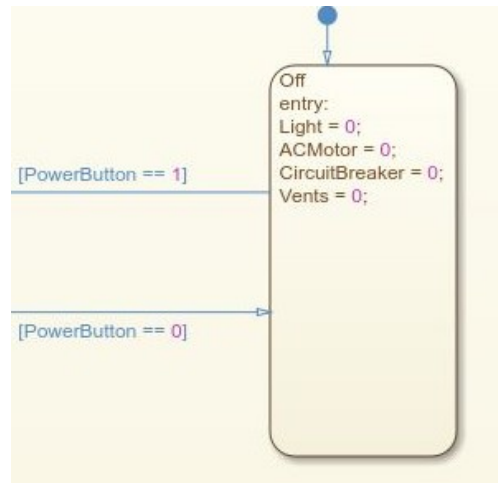


Figure 6: OFF State

4 Control

Another control method is presented which uses a fuzzy logic controller. Fuzzy logic is a powerful problem-solving methodology with a myriad of applications in embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. Fuzzy logic resembles human decision-making with its ability to work from approximate data and find precise solutions. Unlike classical logic which requires a deep understanding of a system, exact equations, and precise numeric values. The main building units of an FLC are a fuzzification unit, a Fuzzy logic reasoning unit, a knowledge base, and a defuzzification unit. Defuzzification is the process of converting inferred Fuzzy control actions into a crisp control actions.

4.1 Designing the Controller

The fuzzy logic controller consists of 2 crisp inputs and 1 crisp output of type mamdani type-I.

4.1.1 Temperature

The first input is the temperature which has a trapezoidal membership function presented in Figure 7. The temperature membership function uses 5 fuzzy sets presented in the following table provided by ?:

Fuzzy Set	Range
Too Cold	0 to 10 C
Cold	5 to 20 C
Warm	15 to 30 C
Hot	25 to 40 C
Too Hot	35 to 45 C

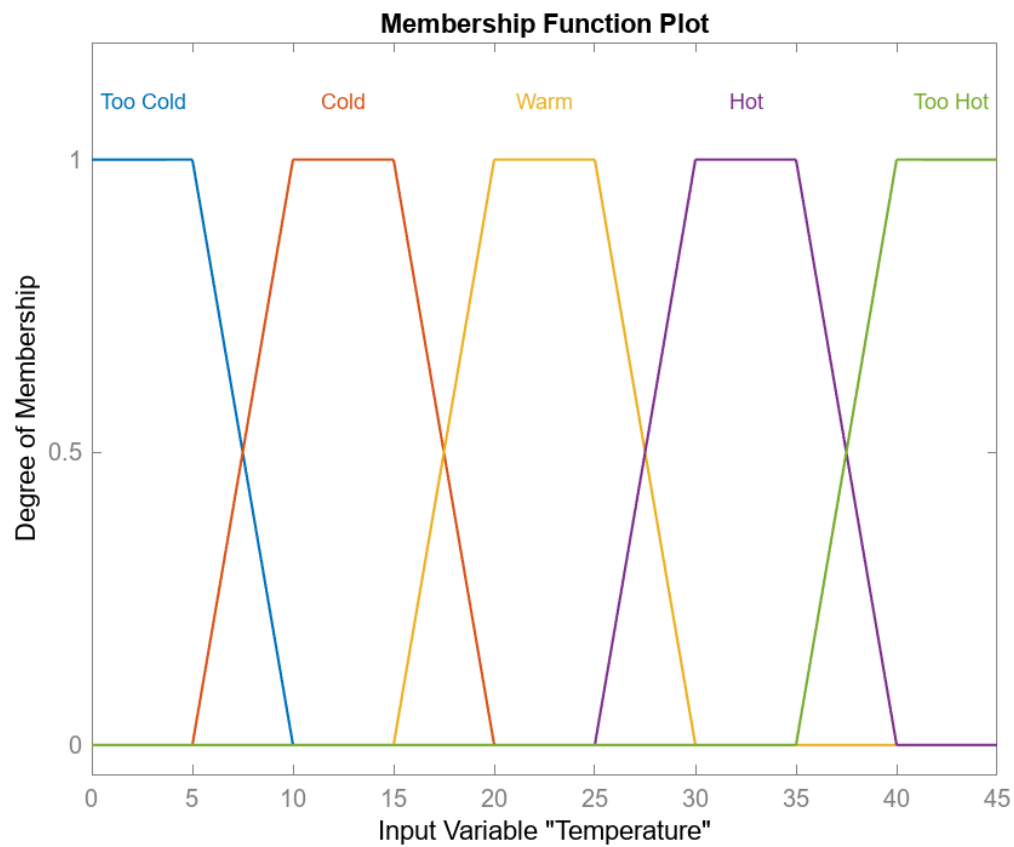


Figure 7: Temperature Membership Function in degrees Celsius

4.1.2 Humidity

The second membership function which is the trapezoidal humidity membership function is presented in the following table and in Figure 8 from ?.

Fuzzy Set	Range
Dry	0 to 30 %
Suitable	20 to 70 %
Wet	60 to 100 %

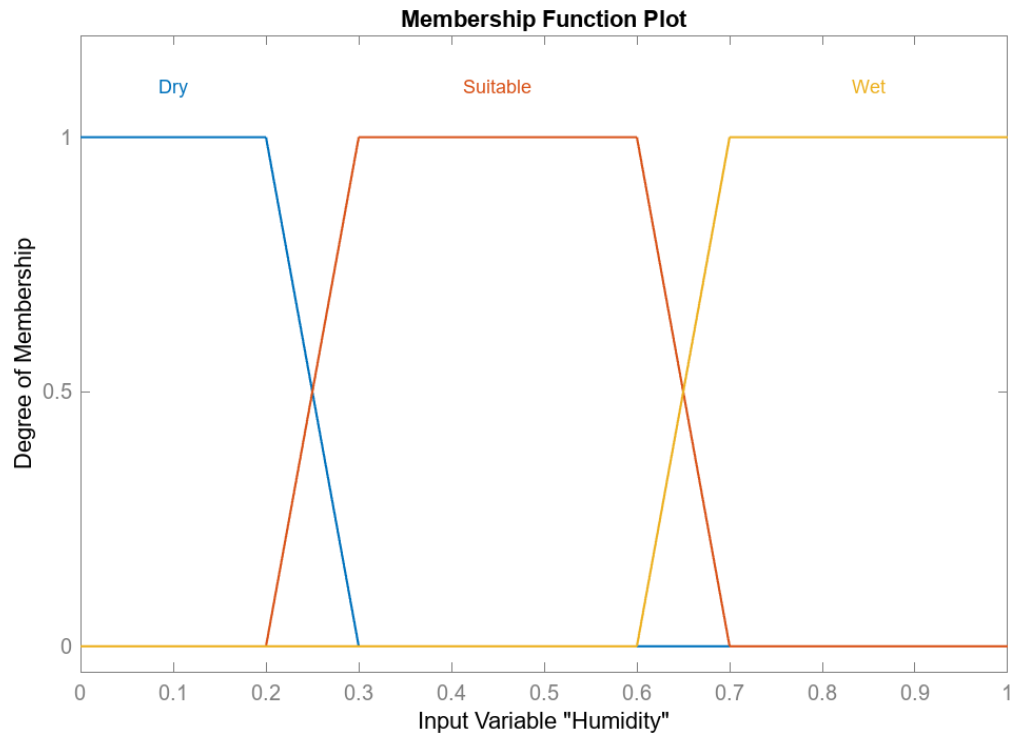


Figure 8: Humidity Membership Function in Percentage

4.1.3 Compressor Speed

The output which is the percentage of the compressor speed also needs to be fuzzified. The trapezoidal membership function is presented in the following table and in Figure 9.

Fuzzy Set	Range
Off	0 to 10 %
Low	0 to 30 %
Medium	20 to 60 %
High	50 to 100 %

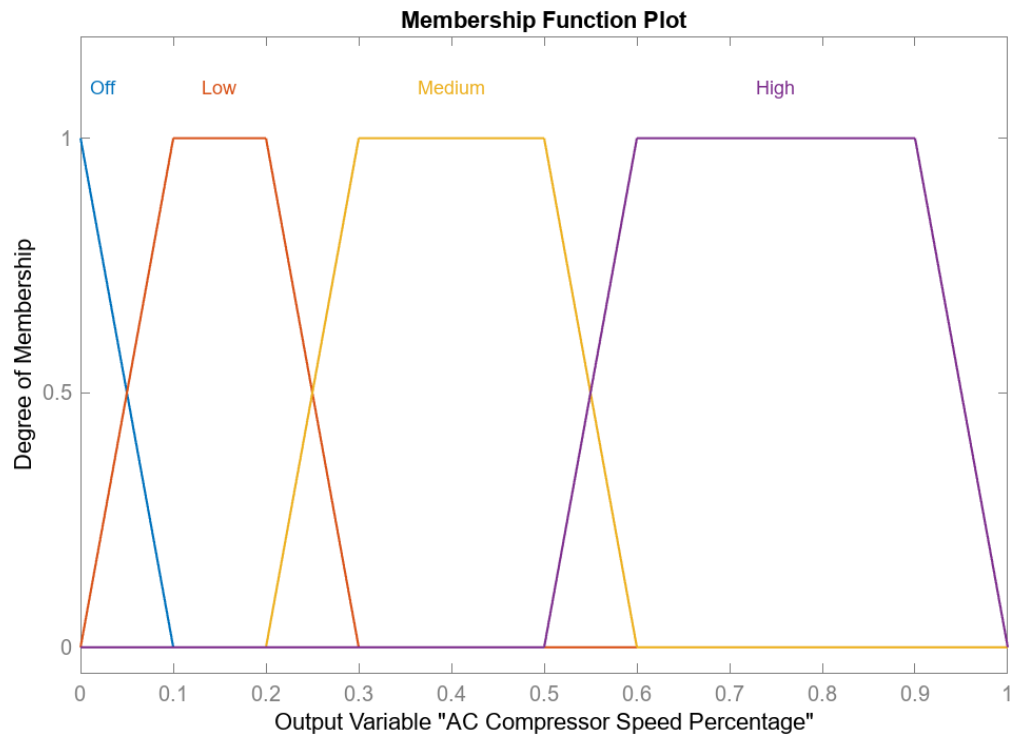


Figure 9: Compressor Speed Membership Function in Percentage

Figure 10 presents the fuzzy logic inference system with both fuzzified inputs and outputs. Next, the set of rules or linguistic variables is presented in if-else conditions in Figure 11 which consists of 15 rules joined by an AND between the 2 inputs and have an equal weight.

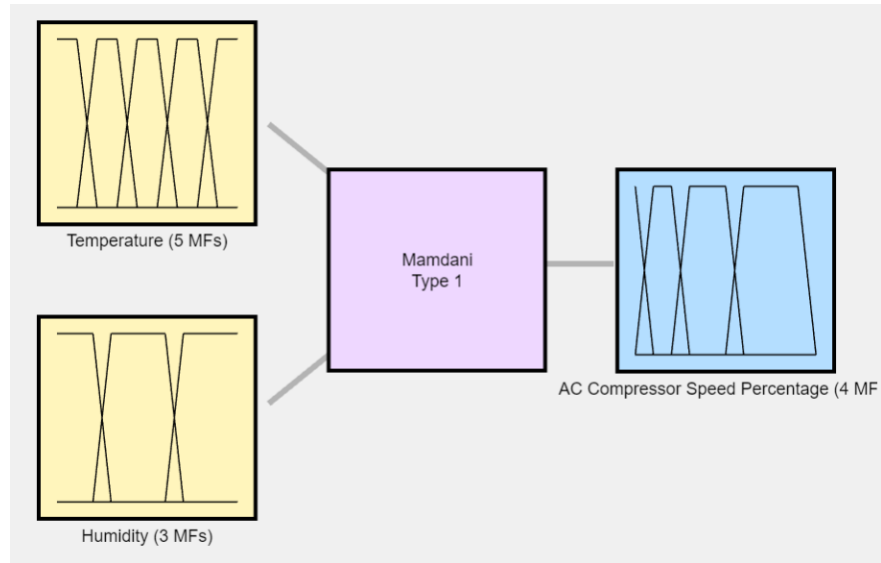


Figure 10: The fuzzy Inference System

	Rule	Weight
1	If Temperature is Too Cold and Humidity is Dry then AC Compressor Speed Percentage is Off	1
2	If Temperature is Too Cold and Humidity is Suitable then AC Compressor Speed Percentage is Off	1
3	If Temperature is Cold and Humidity is Dry then AC Compressor Speed Percentage is Low	1
4	If Temperature is Cold and Humidity is Suitable then AC Compressor Speed Percentage is Off	1
5	If Temperature is Warm and Humidity is Dry then AC Compressor Speed Percentage is Medium	1
6	If Temperature is Warm and Humidity is Suitable then AC Compressor Speed Percentage is Medium	1
7	If Temperature is Hot and Humidity is Dry then AC Compressor Speed Percentage is Medium	1
8	If Temperature is Hot and Humidity is Suitable then AC Compressor Speed Percentage is Medium	1
9	If Temperature is Hot and Humidity is Wet then AC Compressor Speed Percentage is High	1
10	If Temperature is Too Hot and Humidity is Dry then AC Compressor Speed Percentage is Medium	1
11	If Temperature is Too Hot and Humidity is Suitable then AC Compressor Speed Percentage is High	1
12	If Temperature is Too Hot and Humidity is Wet then AC Compressor Speed Percentage is High	1
13	If Temperature is Warm and Humidity is Wet then AC Compressor Speed Percentage is Medium	1
14	If Temperature is Too Cold and Humidity is Wet then AC Compressor Speed Percentage is Low	1
15	If Temperature is Cold and Humidity is Wet then AC Compressor Speed Percentage is Medium	1

Figure 11: Rule Base

The fuzzy logic controller uses the inference rules and the fuzzified inputs and outputs then perform defuzzification

to output a result suitable to the constraints. The defuzzification process is the centroid method. To represent this output, a control surface is plotted in MATLAB in Figure 12.

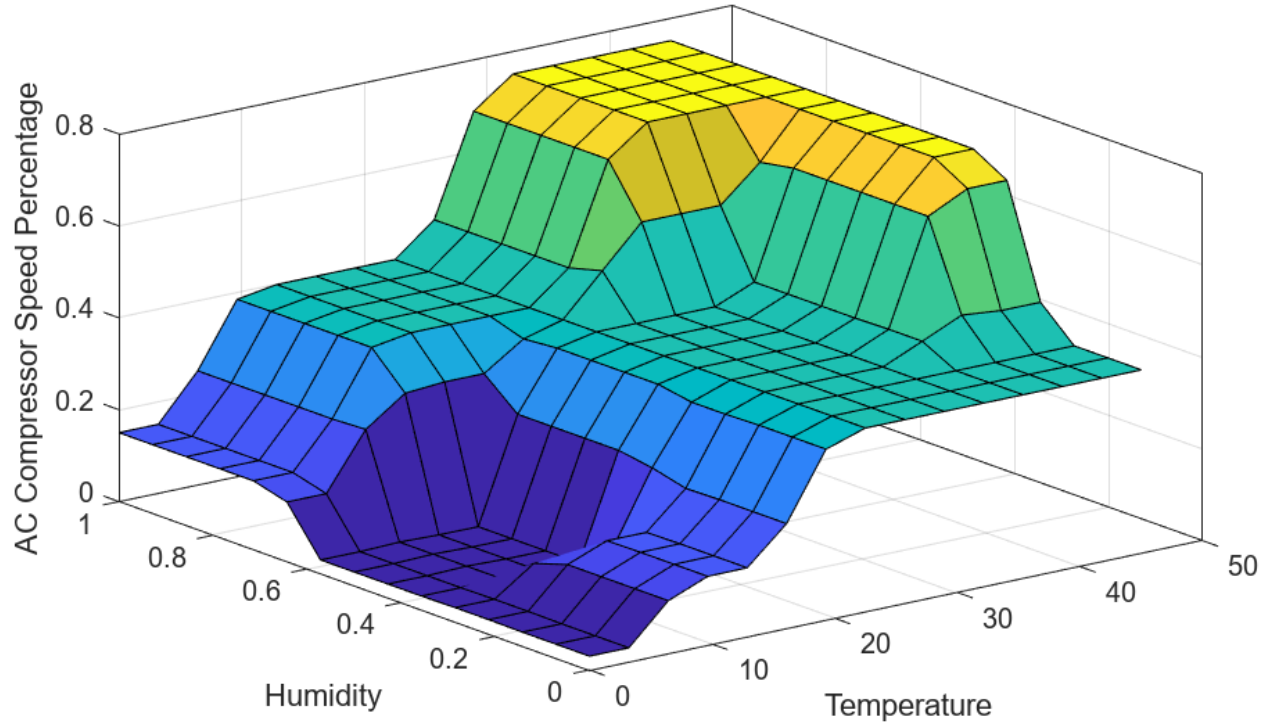


Figure 12: Control Surface showing the Temperature on x-axis, Humidity on y-axis, and the Compressor Speed Percentage on the z-axis.

Next the Simulink function is created and placed inside the AC actuation state. The function consists of the inputs from the sensors, and output to the compressor speed, the mux to group the inputs for the controller, the fuzzy logic controller, and finally a MATLAB function to map the out compressor speed percentage to the corresponding compressor speed.

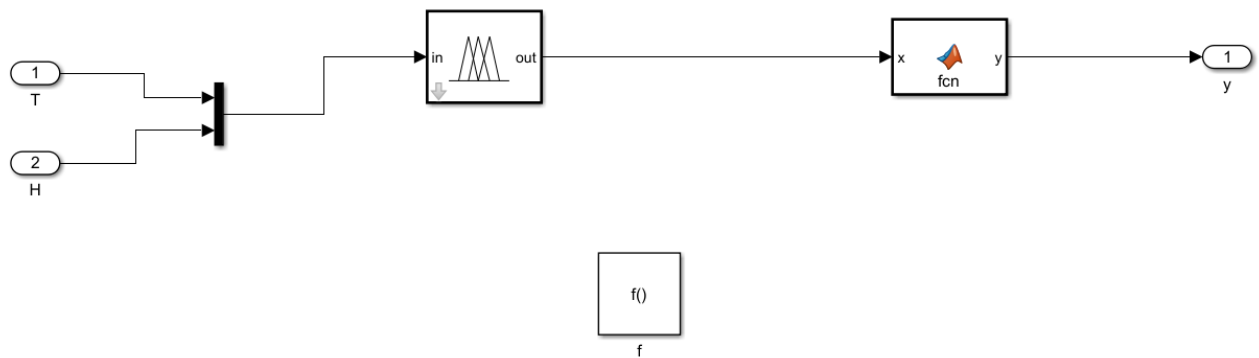


Figure 13: Simulink Model for the Actuation State

The range of the compressor speed is mapped from 1725 to 3450 rpm. When the temperature decreases the compressor speed should be high and vice versa.

5 Design of Hybrid StateChart

5.1 Simulink Model

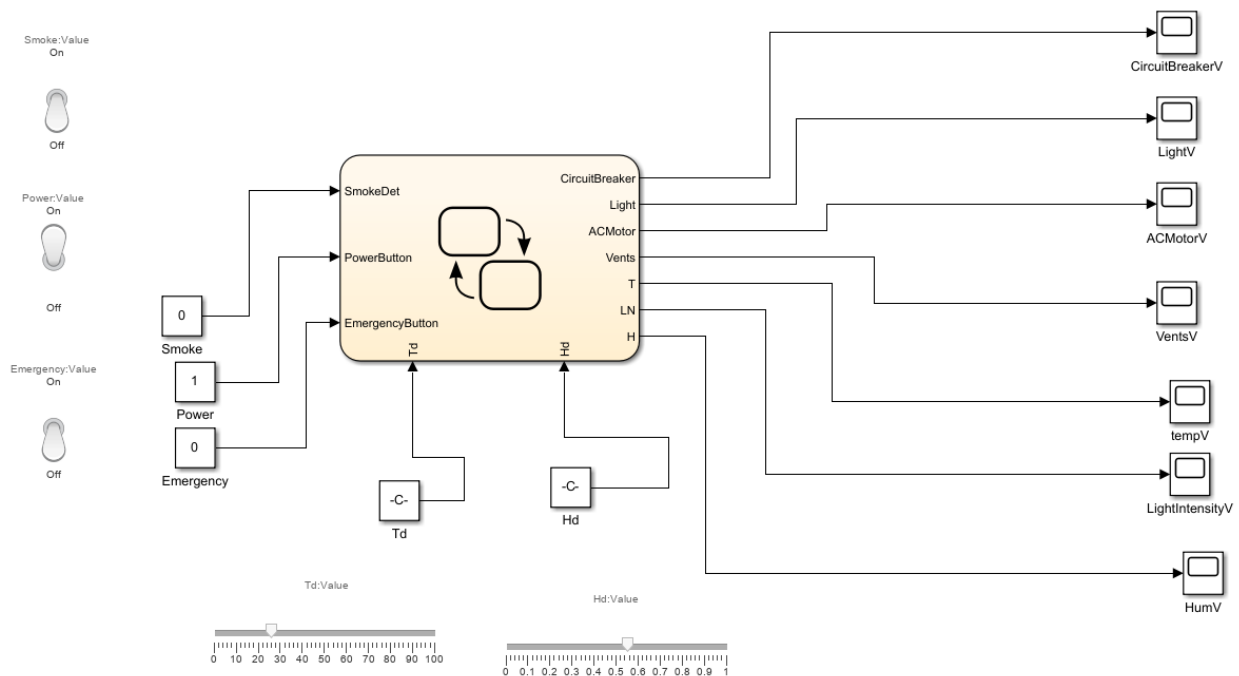


Figure 14: Simulink Model

In figure 14, we represented the power switch, emergency, switch, and smoke detector as the switches on the left side of the model, and the temperature sensor and light sensor as sliders with varying values, in which the temperature varies from 0 degrees to 50 degrees Celsius, and the light sensor varies from 0 to 1000. Then we tested different values for the sensors to visualize the output plots of the light and AC. When we tried a random value (starting from zero) for the light sensor, the output was almost immediate as shown in this plot (figure 19). For the AC motor, we did the same and chose a random temperature value (starting from zero), and the motor reached the required speed in a good response rate, however there was an overshoot for a small period of time, then it converged as shown in this plot (figure 20).

5.2 Hybrid StateChart

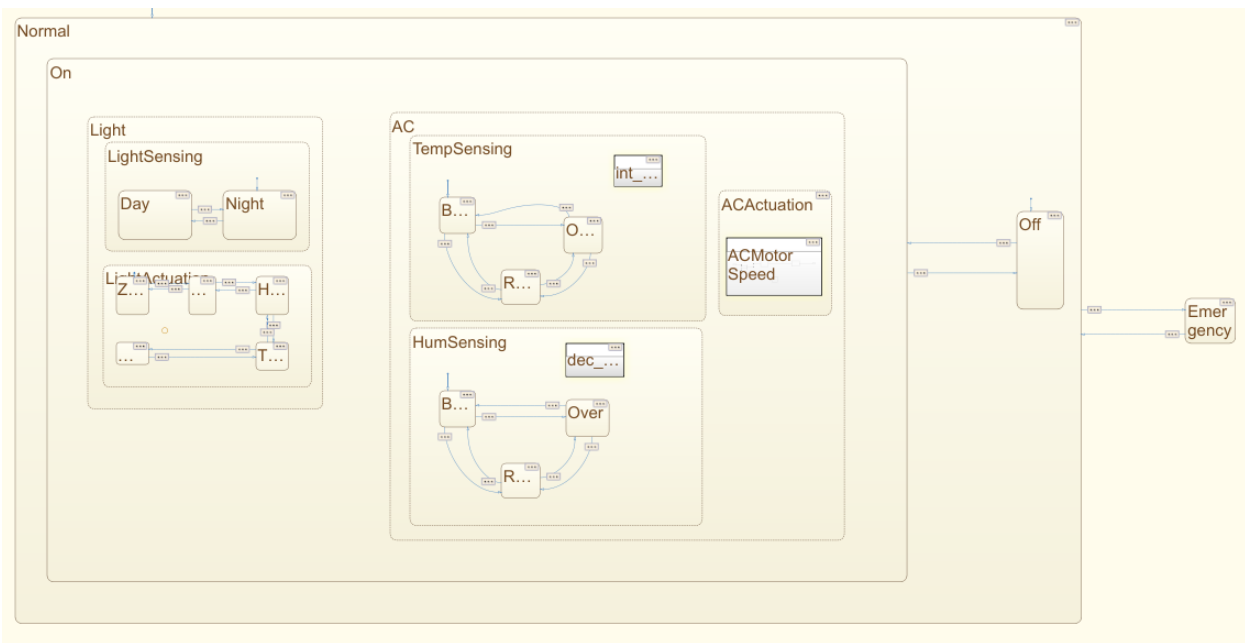


Figure 15: Hybrid StateChart Model

5.2.1 Light Sensor

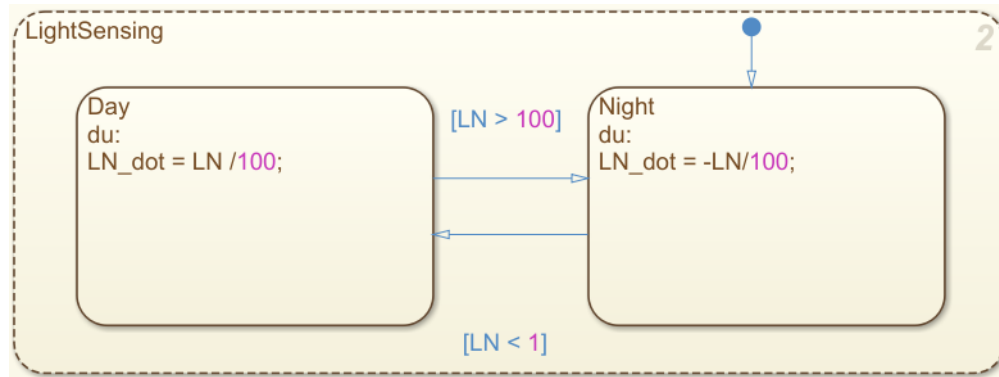


Figure 16: Light Sensor

The light sensor (Figure 16) takes light intensity from the sunlight as input (continuous variable), so when there is still sunlight, the output (Light Bulbs) will vary. If it is night, the light bulbs will be at full intensity, since there is no sunlight.

5.2.2 Temperature Sensor

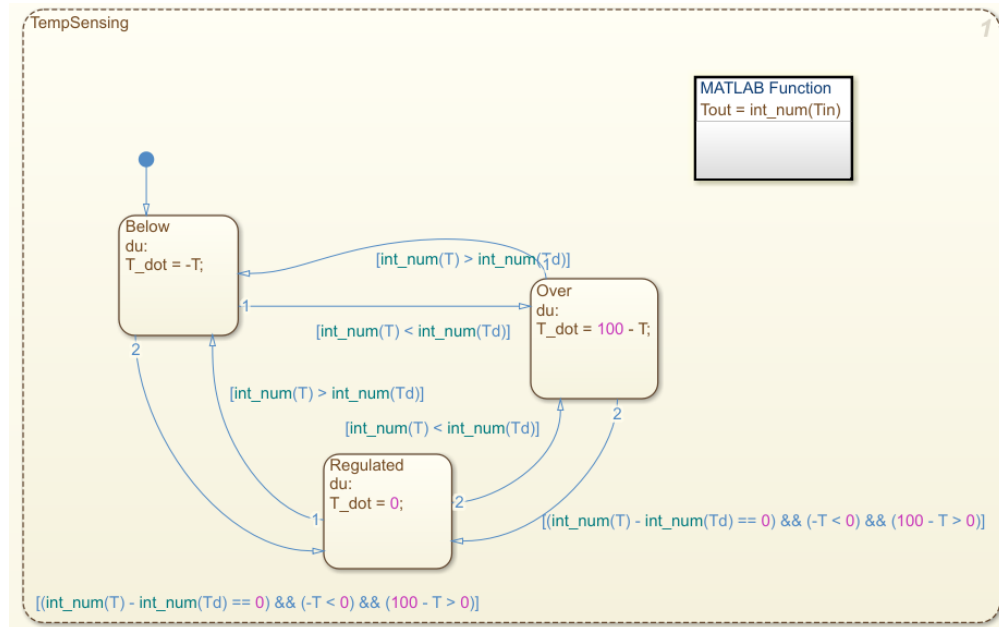


Figure 17: Temperature Sensor

The temperature (Figure 17) takes the reading from the temperature sensor, and it keeps fluctuating (Zeno Behavior) until the sensed temperature reaches the desired temperature. Accordingly, the AC Motor changes its speed.

5.2.3 Humidity Sensor

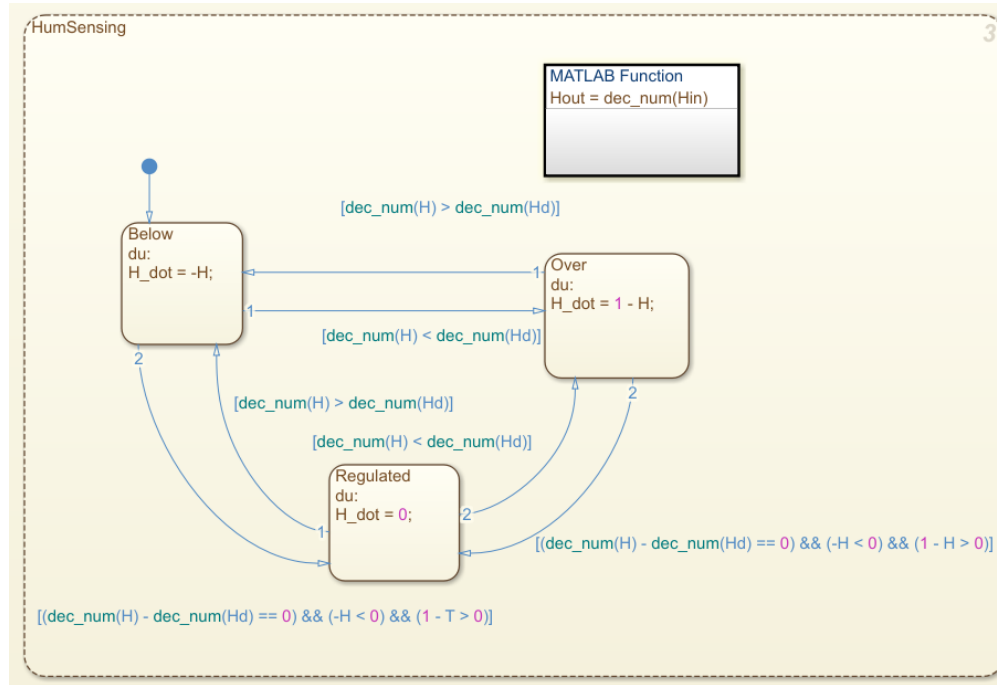


Figure 18: Humidity Sensor

The humidity (Figure 18) takes the reading from the humidity sensor, and it keeps fluctuating (Zeno Behavior), as the temperature sensor, until the sensed humidity reaches the desired humidity. Accordingly, the motor changes its speed.

5.3 Results



Figure 19: Output Light Plot

The light bulbs output (Figure 19) reached the desired illumination according to the sensed light intensity.



Figure 20: Output AC Motor Speed Plot

The AC Motor speed (Figure 20) reached the desired speed according to the desired temperature and humidity.

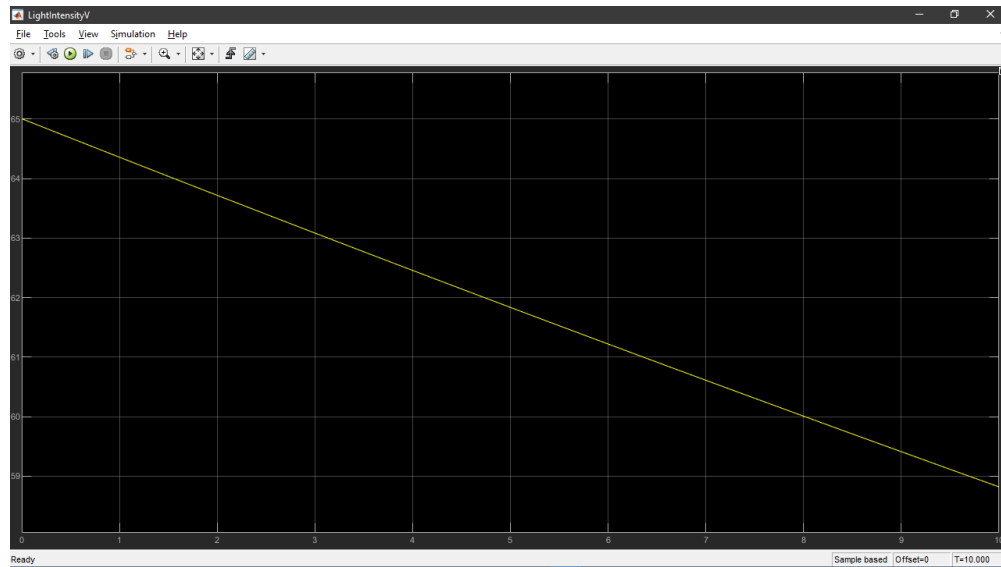


Figure 21: Output Light Intensity Plot

The output light intensity sensor (Figure 21) is the sensed light from the sunlight from outside the warehouse.

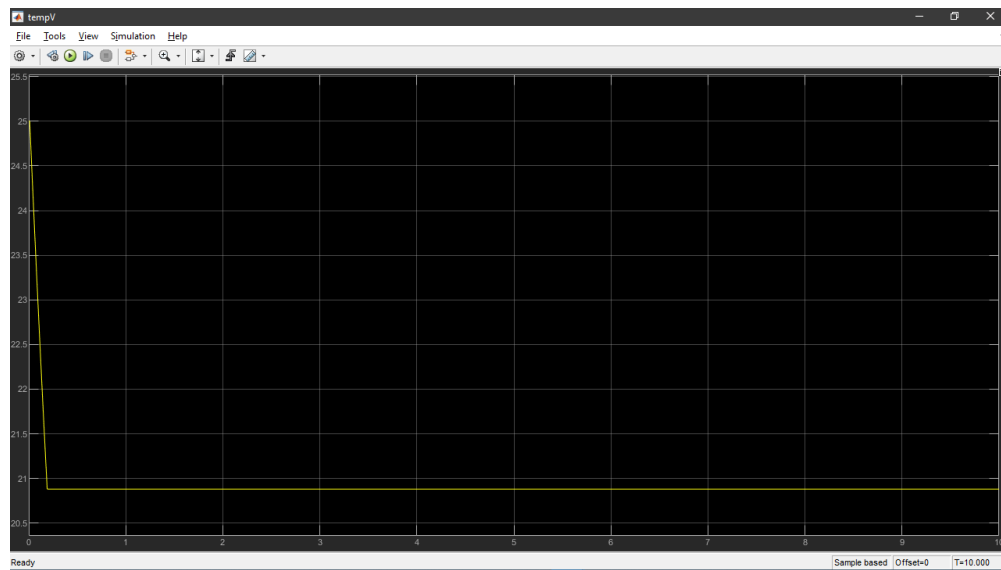


Figure 22: Output Temperature Plot

The output temperature sensor (Figure 22) is the sensed temperature of the warehouse.

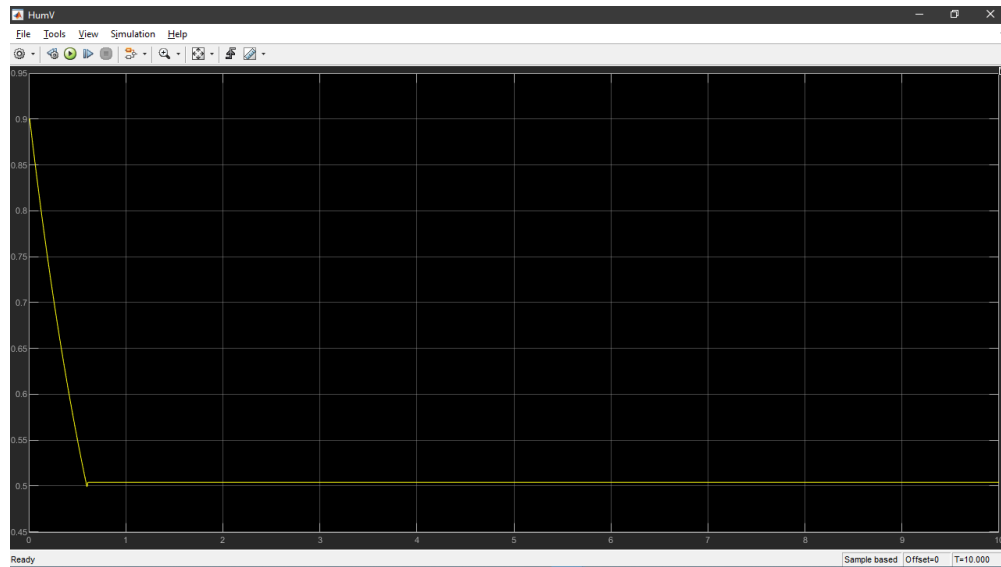


Figure 23: Output Humidity Plot

The output humidity sensor (Figure 23) is the sensed humidity of the warehouse.

5.4 PiL

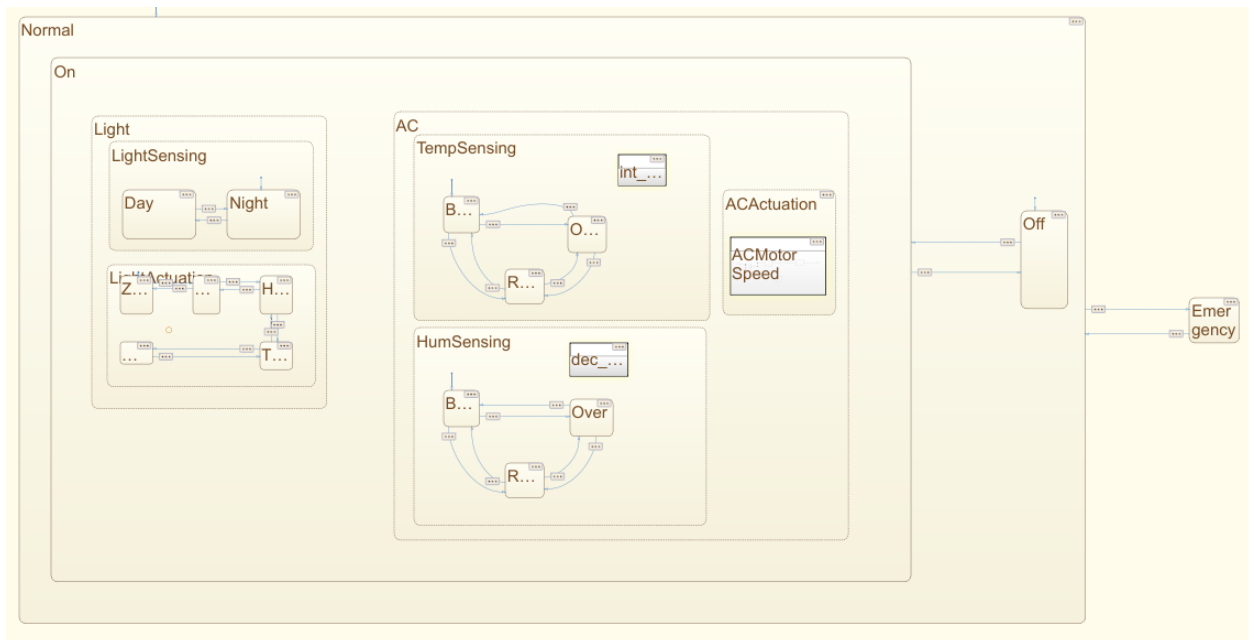


Figure 24: Hybrid StateChart Model

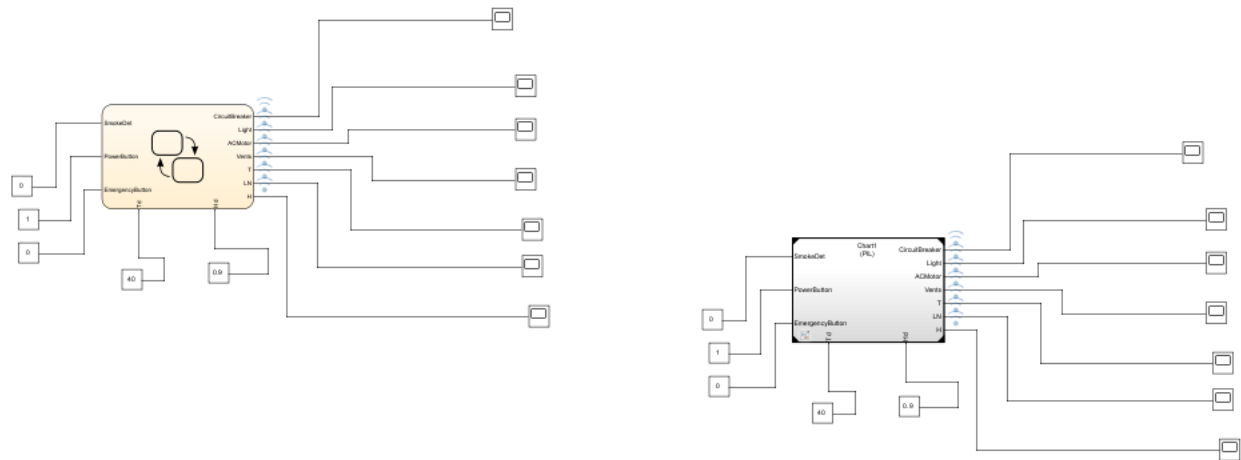


Figure 25: Hybrid StateChart Model (Simulink and PiL)

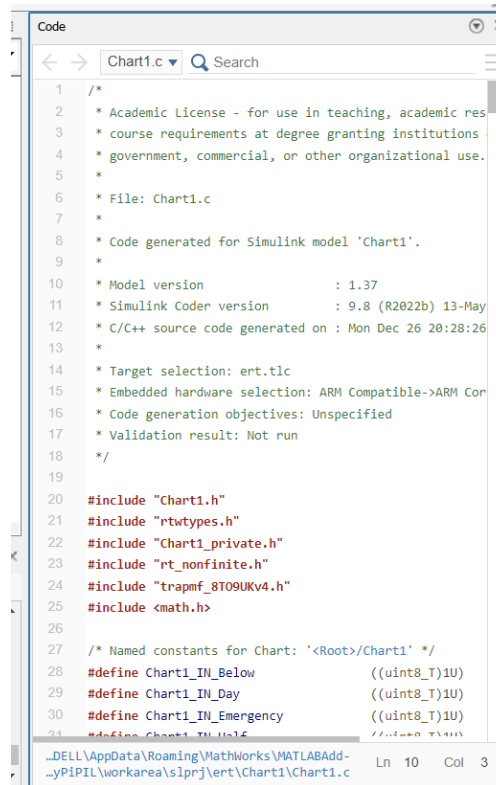
```

08:28 PM: Simulation, 0 2 6
ons\toolbox\raspberrypi\workarea\slprj\ert\chart1\coderassumptions\lib\chart1_ca.mk ...
### Building 'Chart1_ca': make -j$((nproc)+1) -Otarget -f Chart1_ca.mk all
### Using toolchain: GNU GCC Embedded Linux
### Creating 'C:\Users\DELL\AppData\Roaming\MathWorks\MATLABAdd-Ons\Toolboxes\RaspberryPiPiL\workarea\slprj\ert\Chart1\pil\Chart1.mk' ...
### Building 'Chart1': make -j$((nproc)+1) -Otarget -f Chart1.mk all
### Starting application: 'slprj\ert\Chart1\pil\Chart1.elf'
### Launching application Chart1.elf...
### Host application produced the following standard output (stdout) and standard error (stderr) messages:

Logged signals from the two simulations are compared in Simulation Data Inspector.

```

Figure 26: Diagnostic View



```

1  /*
2  * Academic License - for use in teaching, academic res
3  * course requirements at degree granting institutions
4  * government, commercial, or other organizational use.
5  *
6  * File: Chart1.c
7  *
8  * Code generated for Simulink model 'Chart1'.
9  *
10 * Model version          : 1.37
11 * Simulink Coder version  : 9.8 (R2022b) 13-May
12 * C/C++ source code generated on : Mon Dec 26 20:28:26
13 *
14 * Target selection: ert.tlc
15 * Embedded hardware selection: ARM Compatible->ARM Cor
16 * Code generation objectives: Unspecified
17 * Validation result: Not run
18 */
19
20 #include "Chart1.h"
21 #include "rtwtypes.h"
22 #include "Chart1_private.h"
23 #include "rt_nonfinite.h"
24 #include "trapmf_8T09UKv4.h"
25 #include <math.h>
26
27 /* Named constants for Chart: '<Root>/Chart1' */
28 #define Chart1_IN_Below ((uint8_T)1U)
29 #define Chart1_IN_Day    ((uint8_T)1U)
30 #define Chart1_IN_Emergency ((uint8_T)1U)
31 #define Chart1_IN_Half   ((uint8_T)1U)
32
33 ..DELL\AppData\Roaming\Mathworks\MATLABAdd-
34 ..y\PiL\workarea\slprj\ert\Chart1\Chart1.c

```

Figure 27: Successful Generation of C Code

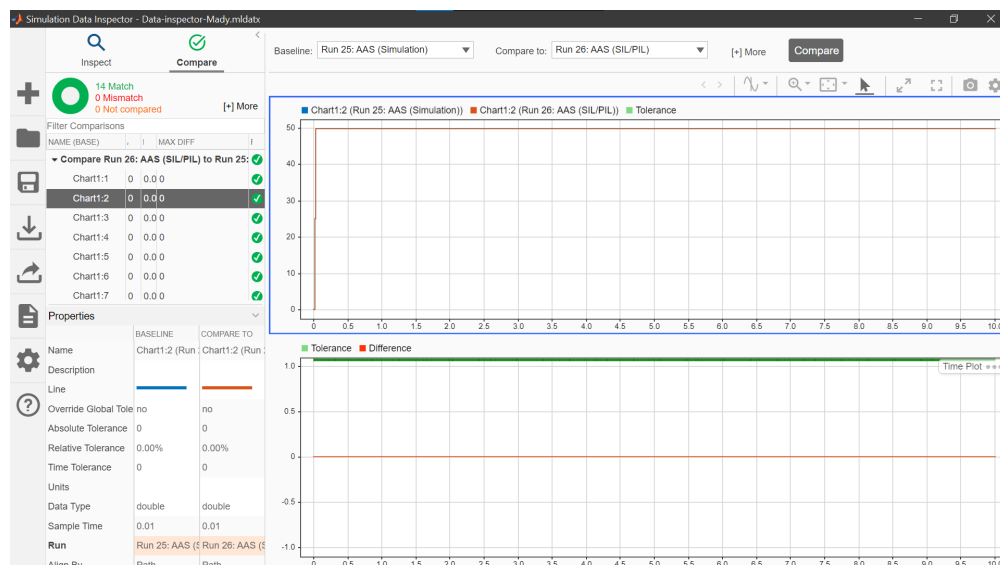


Figure 28: Zero error between the PiL and Simulation from Data Inspector

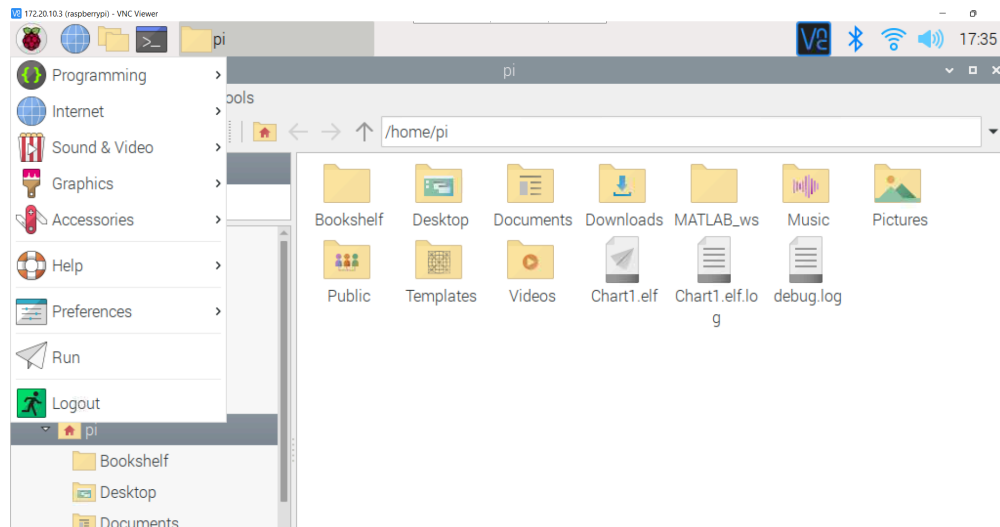


Figure 29: Generated Files on Raspberry Pi

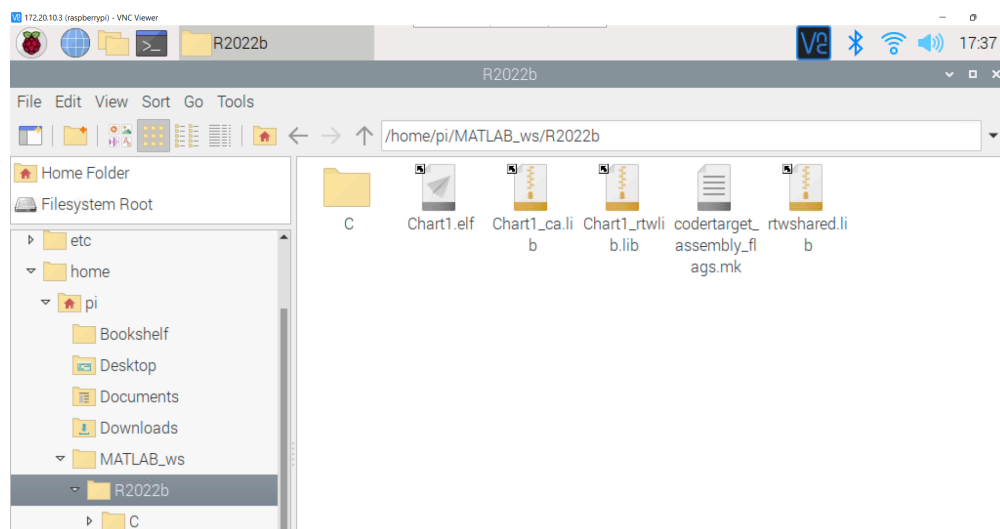


Figure 30: Generated Matlab Files on Raspberry Pi