HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

SENIOR PROJECT 2

DESIGN AND CONSTRUCTION OF SMART BEDROOM USING ARTIFICIAL INTELLIGENCE MODEL

BUI TUAN DAT

STUDENT ID: 19119039

TRAN PHAN BAO KHANG

STUDENT ID: 19119059

MAJOR: **COMPUTER ENGINEERING TECHNOLOGY**

ADVISOR: **TRUONG NGOC SON, PhD.**

Ho Chi Minh City, August 2022

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

SENIOR PROJECT 2

DESIGN AND CONSTRUCTION OF SMART BEDROOM USING ARTIFICIAL INTELLIGENCE MODEL

BUI TUAN DAT

STUDENT ID: 19119039

TRAN PHAN BAO KHANG

STUDENT ID: 19119059

MAJOR: **COMPUTER ENGINEERING TECHNOLOGY**

ADVISOR: **TRUONG NGOC SON, PHD.**

Ho Chi Minh City, August 2022

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 21, 2022

PROJECT ASSIGNMENT

Student name: _____ Student ID: _____

Student name: _____ Student ID: _____

Student name: _____ Student ID: _____

Major: _____ Class: _____

Advisor: _____ Phone number: _____

Date of assignment: _____ Date of submission: _____

1. Project title: _____

2. Initial materials provided by the advisor: _____

3. Content of the project: _____

4. Final product: _____

CHAIR OF THE PROGRAM
(Sign with full name)

ADVISOR
(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 21, 2022

ADVISOR'S EVALUATION SHEET

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Major: .....

Project title: .....

.....

Advisor: .....

EVALUATION

1. Content of the project:

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: (Excellent, Good, Fair, Poor)

.....

6. Mark:…….(*in words*:.....)

Ho Chi Minh City, month day, year

ADVISOR

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 21, 2022

PRE-DEFENSE EVALUATION SHEET

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Major: .....

Project title: .....

.....

Name of Reviewer: .....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark:…….(*in words*:.....)

Ho Chi Minh City, month day, year

REVIEWER

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Student name: ..... Student ID: .....

Major: .....

Project title: .....

.....

Name of Defense Committee Member:

.....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

5. Mark:…….(*in words*:.....)

Ho Chi Minh City, month day, year

COMMITTEE MEMBER

(Sign with full name)

DISCLAIMER

We declare that this dissertation is our work, guided by Associate Professor Ph.D. Truong Ngoc Son. The statements made in the thesis are the results of serious and independent research by the author himself based on researching and researching scientific documents.

The thesis ensures objectivity, honesty, and science. The use of references has been specified in the references section. The data and results presented in the project are sincere.

We do not copy or steal the ideas of any individual or organization. If wrong, I will take full responsibility for the evaluation form of the teacher in charge of the subject.

ACKNOWLEDGMENT

After the completion of significant project work, words are not enough to express our feelings about all those who helped us to reach our goal.

First, we take this opportunity to express our deep regards and heartfelt gratitude to our project guide prof. Truong Ngoc Son for his inspiring guidance and timely suggestions in carrying out the project successfully. He has also been a constant source of inspiration for us. Working with him is an opportunity for us to learn more and more.

We would also like to thank all the teachers of the Faculty of Electrical and Electronic Engineering for providing invaluable support and motivation.

We are also grateful to our friends for their help and cooperation throughout this work and during the past week.

Last but not least, we thank our family for their support, patience, motivation, empathy, and understanding while completing our project.

TABLE OF CONTENT

LIST OF FIGURE

LIST OF TABLE

ABSTRACT

Industry 4.0 has brought Engineering Technology in recent years to a whole new level. Typically, artificial intelligence and embedded industries are considered as an industry with many potential applications and practical applications that bring convenience and performance to people's lives today.

In this topic, our team learns about some basics embedded in Raspberry Pi and trains a basic AI model, learns related theories, and performs training with time and temperature data sets.

CHAPTER 1. INTRODUCTION

1.1. Introduction

Today, research, development and construction, and deployment of smarts or auto bedrooms is also a problem of interest to inventors and researchers in order to serve people in life such as parking rooms, library rooms, apartment rooms, closed rooms of households in life, and resorts. Due to its high applicability, researchers have embarked on developing more and more intelligent rooms, serving the needs of economic and social development. We need to be a connection between the "smart brain" and consumer devices such as lights and fans.

Machine Learning or Artificial Intelligence is the foundation that makes rooms smarter and more sophisticated. Here are some concepts of artificial intelligence:

Intelligence in the oxford dictionary is the ability to learn, understand and think.

Artificial Intelligence which word intelligence means created by humans.

Intelligence is the ability to handle situations when new problems arise, and human intelligence can be described so accurately in order to a machine can be created to simulate it.

Thanks to these things, we have the basis of artificial intelligence, which is computing large and enough data to simulate humans.

1.2. Urgency of the topic

Compared to waiting until we get to the bedroom or when we sleep, we will turn on and adjust the fan or air conditioner that can intelligently adjust to the desired temperature, we find it will help us a lot in reducing stress to not having to wake up to adjust the device and we also see that we can protect our health if the weather changes or we accidentally turn on the devices that exceed the environmental conditions at the last time we adjust all devices.

1.3. Objectives of the study

Learn about Python and TensorFlow.

Learn about the operation method such as the connection diagram of the relay, DS18B20 temperature sensor, push button, led, and Raspberry Pi 4.

Bedroom design.

1.4. Research mission

Build bedroom model.

Using Python and TensorFlow for AI and controller programming.

1.5. Object and scope of the study

Raspberry Pi 4 board: here we focus on learning, studying how to install the operating system, using the Raspberry Pi's Gpio pins, and also training the AI model on the raspberry.

Python programming language: we use python language throughout the process, and it is the language used in the article to compile and train AI.

DS18B20 temperature sensor: we will learn how to read the temperature and use the temperature to train AI.

TensorFlow: we will learn TensorFlow at a basic level to understand and apply to the topic.

1.6. Research Methods

To carry out the original idea of the topic, we used the following methods:

- Methods of synthesizing theoretical documents and learned knowledge.
- Test methods.
- Method of consulting the instructor.

1.7. Project Layout

The project layout is divided into six chapters as follows:

Chapter 1: Introduction.

Chapter 2: Theoretical background.

Chapter 3: Hardware Design.

Chapter 4: Software Design.

Chapter 5: Experimental results.

Chapter 6: Conclusion and development direction

CHAPTER 2: THEORETICAL BACKGROUND

2.1. Raspberry Pi 4 Model B

2.1.1. Introduction to Raspberry Pi 4 Model B

Raspberry Pi is a small computer that integrates much more powerful hardware capable of running the operating system and installing many applications on Raspberry Pi.

Raspberry Pi 4 Model B is the latest product in Raspberry Pi computers.

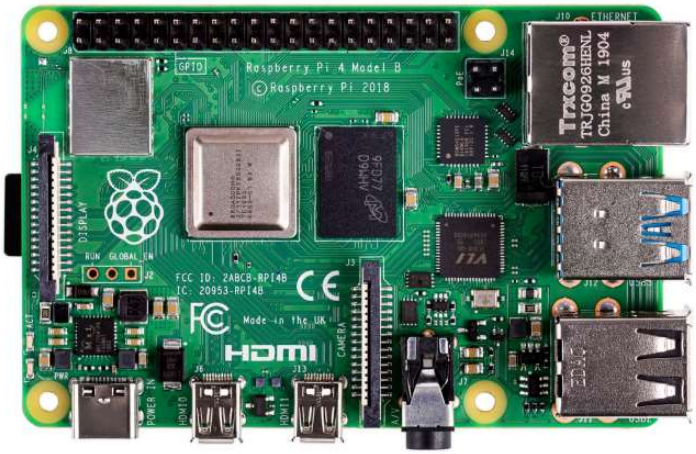

Figure. 1: Raspberry Pi 4.

Raspberry Pi 4 brings a lot of upgrades in hardware in every way. These include CPU upgraded performance 3 times, Wi-Fi ac + Bluetooth 5.0 wireless network, and RAM options up to 4 GB for $ 35 (1 GB RAM).

The 4th generation circuit board structure is still similar to the previous generations of Raspberry Pi. The new features are listed below:

- Power supply: use USB-C port instead of micro USB with 5V-3A adapter, can use Nokia or Google Pixel charger, provide 1.2A power supply to peripheral devices using USB-A.
- Monitor connection: use 2 ports of micro HDMI (type-D) instead of the previous full-size HDMI port. VideoCore GPU supports OpenGL ES 3.0, supports video up to 4K60Fps.
- Communication: USB is connected via PCIe Gen 2 protocol, providing a total bandwidth of 4 Gbps for 4 ports 2x USB 3.0, and 2x USB 2.0. PoE HAT compatible 1 Gbps LAN port.
- Wireless connection: using dual-band 802.11ac Wi-Fi, Bluetooth 5.0.
- The hardware of the Raspberry Pi 4 operates on the Broadcom BCM2711 SoC platform with a processor quad-core 1.5 GHz core Cortex-A72. The platform supports LPDDR4 RAM, and the system's storage is provided via a microSD card

port.

2.1.2. Raspberry pi 4 configuration

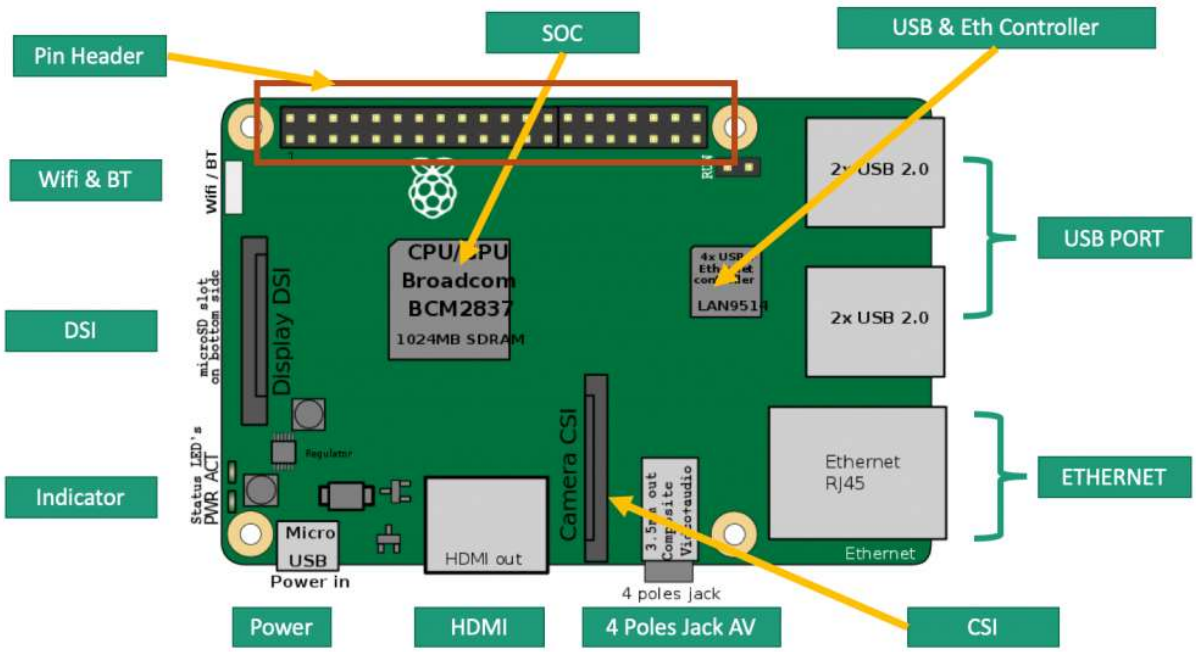

Figure. 2: Brief component of Raspberry Pi4 Model B..

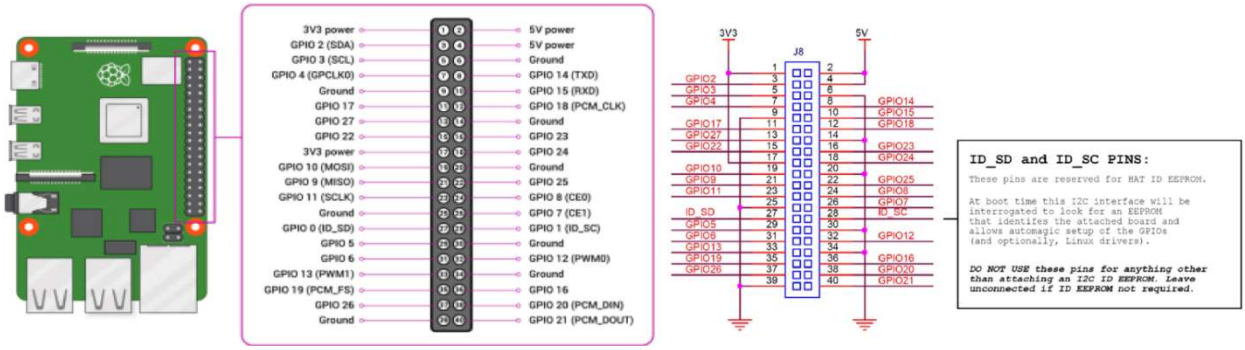

Figure. 3: Pinout of Raspberry Pi 4 Model B.

2.1.3. Operating system on Raspberry Pi

After the introduction and structure of the Raspberry Pi, we move on to the part to make the Pi work. Unlike other micro-controllers that can run immediately after loading the control program. The Important of Raspberry Pi is that requires an operating system to work. That is also an advantage of the Pi because it allows users to take advantage of a lot of software and peripherals to program complex applications quickly.

2.2. Python programming language

2.2.1 Introduction

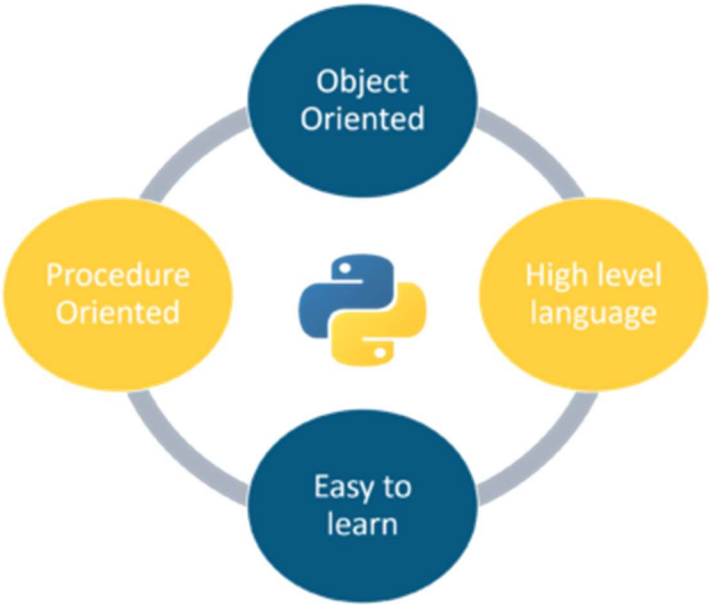

Figure. 4: Python programming in Information Technology.

Python is an interpreted programming language released by Guido van Rossum in 1990. The interpreted language here means that it directly runs code line by line, if there is an error in the program code, it will stop running.

Python is in steady development for basically its entire existence from around 2010 year it started on a development trajectory that soon allowed it to be promoted with other top programming languages recently, such as Java and JavaScript.

Today, according to statistics, Python is widely used in all IT professions.

2.2.2 Features of Python programming language

Guido van Rossum defined his goals for Python in 1999 and it really holds true to this day:

- A language-use easy and intuitive also powerful like the language of significant competitors.
- Open Source so anyone can contribute to its development clearly.
- Easy to understand code use English for the starter.
- Suitable for everyday tasks, allowing short development time.

2.3. TensorFlow

2.3.1 What is TensorFlow?

Figure. 5: TensorFlow.

TensorFlow is a widely used open source library in machine learning, which makes it faster and easier to increase speed.

TensorFlow was created and developed by a team of experts at Google, specifically Google Brain. It was created for the primary purpose of being used for research needs and applied in production most effectively.

The First version of TensorFlow was released in November 2015.

2.3.2. Versions

Currently, TensorFlow has two versions, the TensorFlow version1 and TensorFlow version 2. This release contains contributions from many people at Google with the uesful features and improvements.

2.3.3. Keras API

Keras from the position of a python library supporting deep learning frameworks has become the main API recommended by Google in TensorFlow 2. x. The way to write sessions in 1. x versions are also gone, so if you use TensorFlow, you should immediately upgrade to 2.0 and use tf.Keras.

There are three ways to create a neural network model:

Sequential API: simple to write, but will not build shared layers (residual block for example), cannot handle multiple inputs/outputs, so cannot build some models like Resnet, MVCNN, etc.

Functional API: writing is somewhat similar to TensorFlow graph version 1. x, but still can create complex models, and layers that can be shared simply. In addition, all Sequential models can be created using the Functional model.

Model subclassing: writing is somewhat similar to PyTorch subclassing. Capable of writing complex models as well as other Functional model capabilities.

2.3.4. Model training APIs

We have some method use in the model:

Compile method: Configure the model to train and it takes the following arguments:

optimizer: here is the String or optimizer instance.

loss: this is the Loss function. Can be a string (name of loss function). The loss function ought to return a tensor float. In terms of a custom Loss instance being used and reduced the value is set to None.

metrics: List of metrics evaluated by the model during training and testing.

loss_weights: often Python floats or lists to weight the loss contributions of other model outputs.

weighted_metrics: List of metrics evaluated and weighted in sample_weight or class_weight during training and testing.

run_eagerly: type use Bool. If the default is False. If True, the Model's logic will not be covered in a tf. function. Recommend leaving this None unless your Model cannot run inside a tf. function. run_eagerly=True is not supported when you use this statement "tf.distribute.experimental.ParameterServerStrategy".

steps_per_execution: It is Integer. If defaults to 1. The number of batches which to run during each tf. function call.

jit_compile: If True, the compiler will compile the model training step with XLA. In detail, for machine learning, XLA is an optimizing compiler.

*****kwargs***: Arguments will be supported which for only backward compatibility.

Fit method: Train the model for a fixed number of epochs (iterations on a dataset).

Its argument is:

x: Input data (NumPy array, a list arrays,tf. data ...).

y: Target data (mostly get from x, it can be Numpy arrays or TensorFlow tensor(s)

batch_size: real number or None.

epochs: real numbers.

verbose: here can be 'auto', 0.1 or 2. When 0 is silent, 1 is progress bar, 2= one line per epoch.

callbacks: applied throughout the training of the list of callbacks.

validation_split: here is a Float between 0 and 1. Part of the training data will be used as validation data. The model will isolate this part of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. The validation data is selected from the final samples in the x and y data provided, before scrambling. This argument is not supported if and only if x is a dataset, generator.

validation_data: Data to evaluate the loss and any model metrics at the end of each epoch In terms of the model, it will not be trained on this data.

shuffle: it has 2 cases as boolean or str. Boolean when shuffling the training data before each epoch and str for 'batch'. Some caveats are that this argument is ignored when x is a generator.

class_weight: optionally maps from integer index to weight value which is float and used to calculate weight for loss function during training.

sample_weight: Numpy array of weights for the training samples to use for the loss function weights. some can pass one direction Numpy of the same length as the input samples (1:1 mapping between weights and samples) or in In the case of temporary data, you can pass a 2D array. An important note is that there is no supported if and only if x is a dataset, generator.

initial_epoch: here pass as Integer.

steps_per_epoch: you can enter an integer or None.

validation_steps: Only involved if validation_data has and is a tf. data dataset.

validation_batch_size: Here is Integer or maybe None.

validation_freq: The argument here is only relevant if validation data is provided.

max_queue_size: The fill argument can be an integer.

workers: is only true when it is an integer.

use_multiprocessing: is a boolean. If True, use process-based threading. If it cannot be specified, use_multiprocessing will default to False.

Predict method:

x: Input samples (A Numpy array, a list of arrays, TensorFlow tensor...)

batch_size: Is integer or None. If not specified, it will default to 32.

verbose: there are cases like “auto”,0, 1, 2 . 0 equal silent, 1 equal progress bar, 2 equal single lines. "auto" defaults 1 for most cases, and 2 when used with ParameterServerStrategy.

steps: total number of steps before prediction complete.

callbacks: applied throughout the training of the list of callbacks.

max_queue_size: Only Integer.

workers: it is an integer.

use_multiprocessing: Boolean. If True, we use process-based threading. If unspecified, use_multiprocessing will default to False.

2.4. Led

Figure. 6: A figure of LED package.

Led (light-emitting diode) has 2 pinouts include of positive, and negative.

Voltage: 3V.

In essence, the LED is a polarized semiconductor device like a normal diode (with one leg being Anode and one leg being a cathode).

The light only lights up when there is currently going from the Anot to the Katot, which means that the Anode must be connected to the positive source and the Katot to the negative source.

Since LEDs can emit light without emitting energy in the form of heat, it is widely used for lighting and information display for devices that need to save energy.

2.5. DS18B20

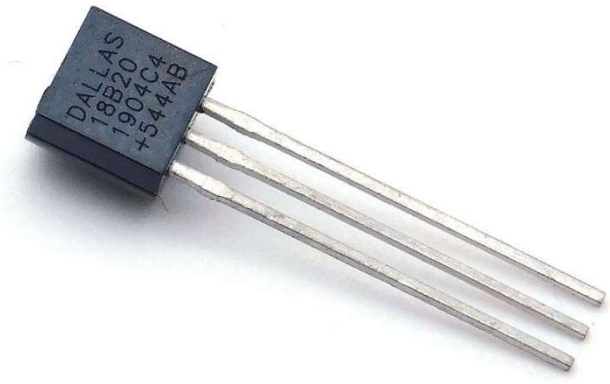

Figure. 7: DS18B20 temperature sensor.

Feature:

- 1-WIRE interface requires only one port pin to communicate.
- Does not require external components.
- Can be powered from the data line. The power supply in range of 3.0V to 5.5V.

– No standby power required Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F.

– Accuracy ±0.5°C from -10°C to +85°C.

– Programmable thermometer resolution which is from 9 to 12 bits.

– Convert 12-bit temperature to digital word in 750 ms (max).

– User definable, non-volatile temperature alarm setting.

– The alarm lookup command identifies and addresses devices whose temperatures are outside the programmed limits (temperature alarm conditions).

– Applications include temperature controls, industrial systems, consumer products, thermometers or any temperature sensitive system.

Pin description

– GND – Ground.

– DQ - Data In/Out.

– VDD - Power Supply Voltage.

– NC - No Connect.

Block diagram

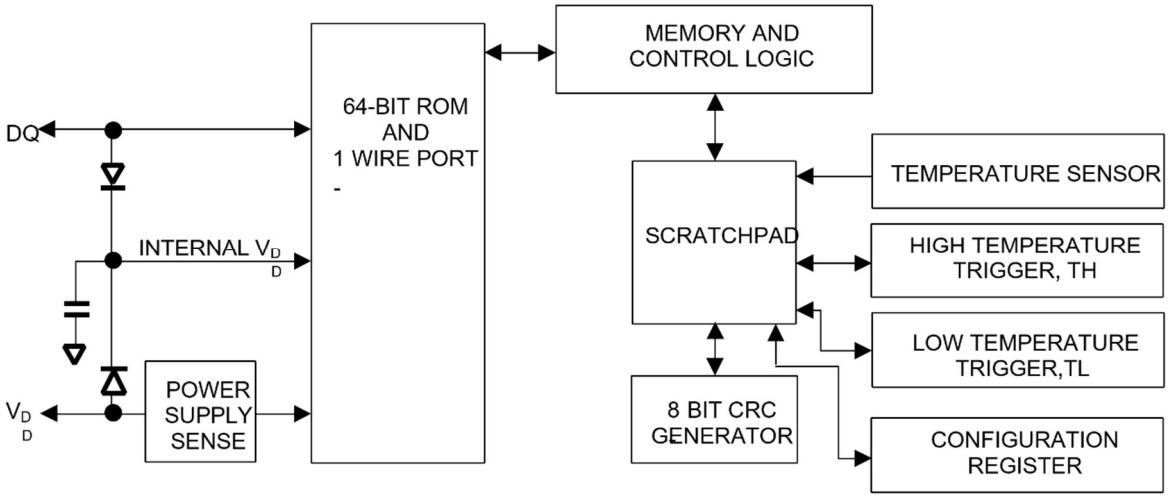

Figure. 8: The block diagram of DS18B20 temperature sensor structure.

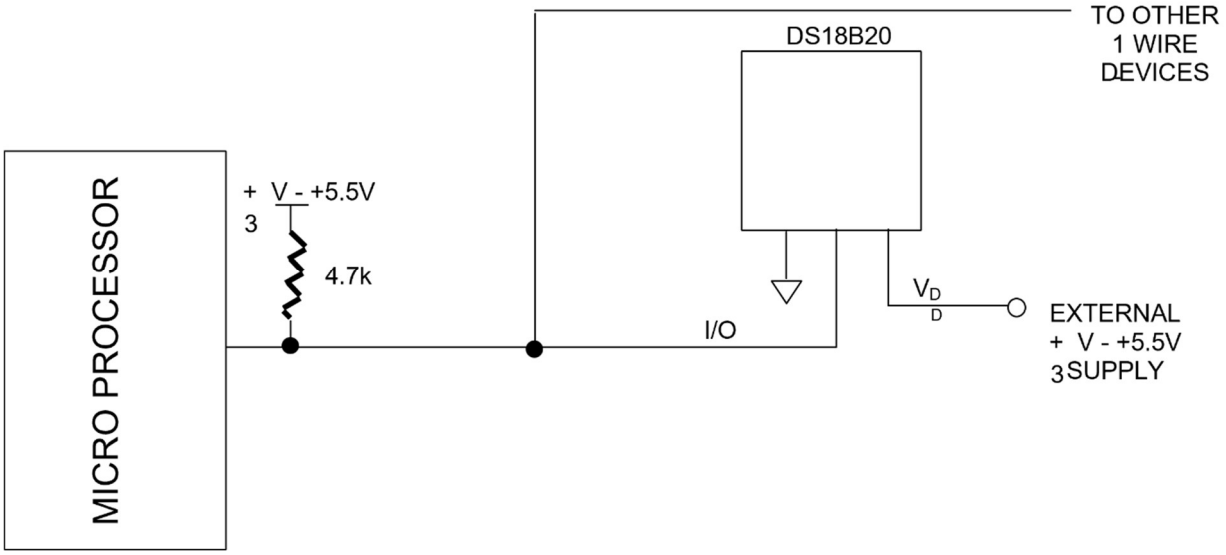

Figure. 9: DS18B20 sensor connect with micro processor.

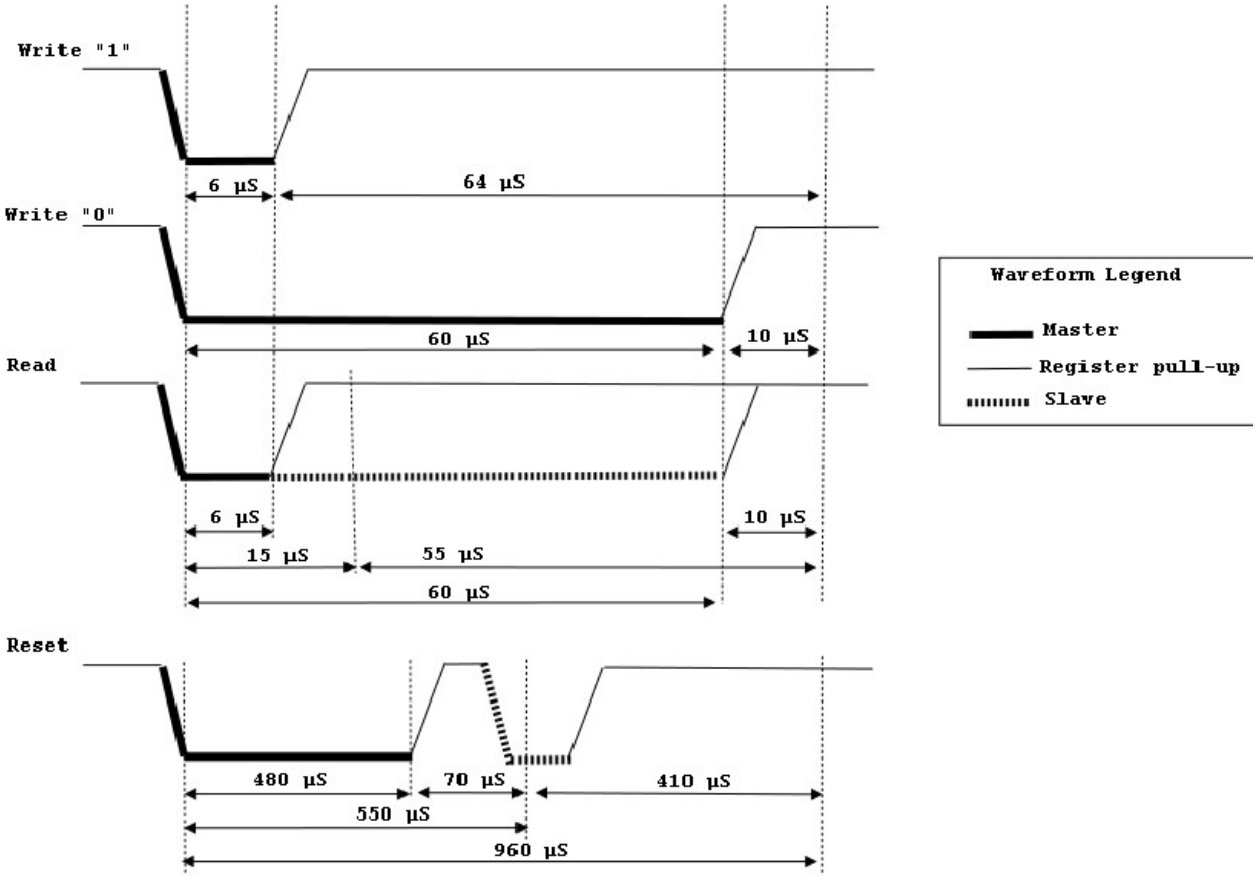

Figure. 10: 1-Wire communication through software.

2.6. LCD16x2

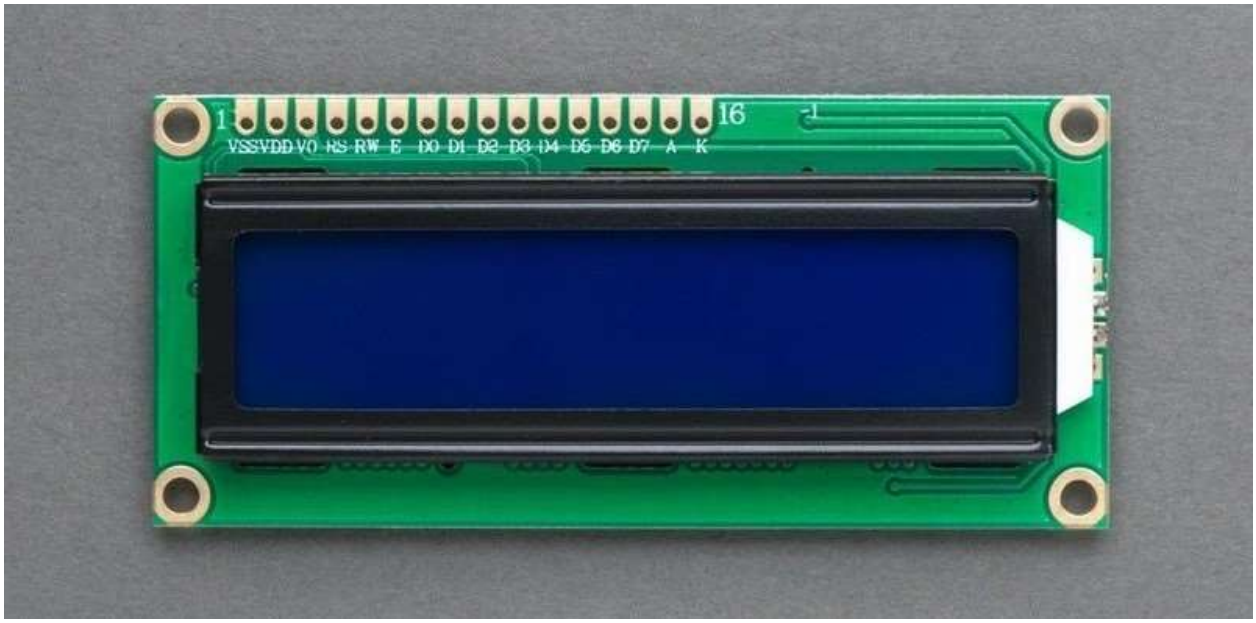

Figure. 11: LCD16x2.

Feature: LCD16x2 is an LCD that can display 2 rows of 16 characters each:

- Display size: 16x2 lines.
- Display color: white and black.
- Communication mode: 8bit or 8 bit.
- Display font size: 5x7 or 5x10.
- Number of pins: 16.
- Operating temperature: 0 to 50 ºC.
- Current Consumption: 2.5 to 4 mA.

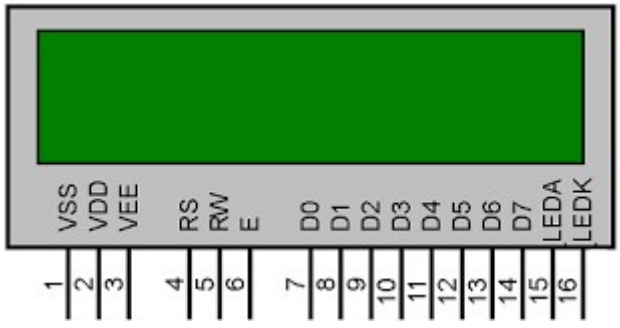

Figure. 12: Pin out of LCD diagram.

The function of the Pins.

Pin	Name	Function of the Pin
1	VSS	Ground pin for LCD, when designing we connect to the microcontroller GND pin.
2	VDD	Power supply pin for LCD, when designing we connect to the 5V source of the microcontroller.
3	VEE	Used to adjust the contrast for LCD
4	RS	Registor Select pin. Use low level when connecting GND, use high level when connecting VDD: – Low level: data bus connects to LCD's instruction register (Write) or connects to LCD's address buffer (Read). – High level: data bus connects to DR data register in LCD.
5	R/W	Read/Write mode selector pin: – Connect level pin "0" to select Reading mode. – Connect level pin "1" to select Writing mode.
6	E	Enable pin. After connecting the data bus signal, an enable pulse is required on this pin to execute instructions.
7	DB0 – DB7	Use 8 pins to receive the signal of the Microcontroller: – 8-bit mode: data transfer on 8 data lines from DB7 to DB0. – 4-bit mode: data transfer on 4 data lines from DB7 to DB4, MSB is DB7.
8	Led + Led –	Turn on the light for the LCD. – Led + connected to VDD. – Led – connected to GND.

Table. 1: Function of the Pins LCD.

2.7. Relay

Figure. 13: Relay module.

Feature:

- Switching capacity available by 10A.
- Simple magnetic relay circuit to meet low cost mass production.
- Selection of plastic materials for high temperature and better chemical solution performance.

Max. operating speed		20 cpm
Initial insulation resistance		Min. 100 M* (at 500 V DC)
Initial breakdown voltage	Between open contects	750 Vrms for 1 min
	Between contact and coil	1,500 Vrms for 1 min
Operate time (at nominal voltage)		Approx. 10 ms
Release time (without diode) (at nominal voltage		Approx. 10 ms
Tenperature rise (at nominal voltage)		Max. 35°C
Shock resitance	Functional	Min. 98 m/s^2
	Destuctive	Min. 980 m/s^2
Vibration resistance	Functional	Approx. 98 m/s^2, 10 to 55 Hz at double amplitude of 1.6mm
	Destructive	Approx. 117.6 m/s^2, 10 to 55 Hz at double amplitude of 2 mm
	Ambient temp	-40°C to +85 °C

Conditions for operation, transport and storage (Not freezing and condensing at low temperature)	Humidity	5 to 85% R.H
Unit weight		Approx. 12g

Table. 2: Feature of Relay.

Schematic of Relay

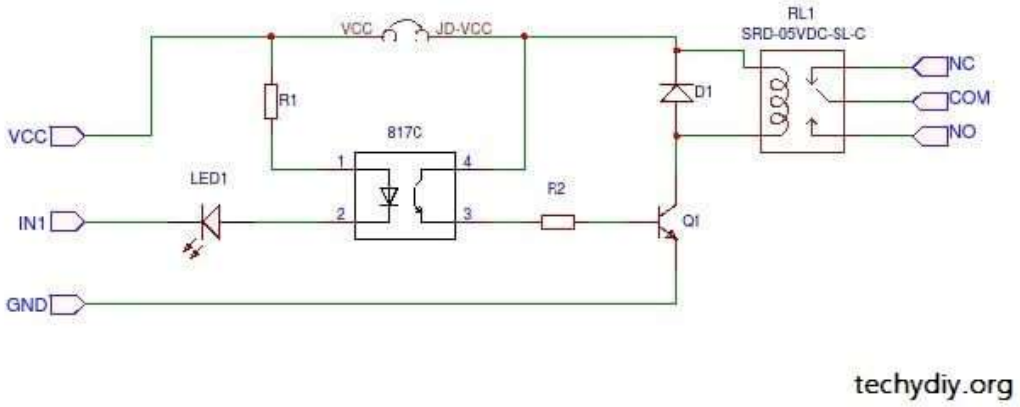

Figure. 14: Schematic of Relay

Relay is an electrical switch that is controlled by a microcontroller signal. The switching process requires a source to perform the function of changing the relay state. The relay connection in the system includes

– Connect DC+battery to 5 VDC . source

– Connect DC-pin to GND–

– Connect the Signal pin to the output of the microcontroller

– NC connected to Power Supply

– Output relay connected load

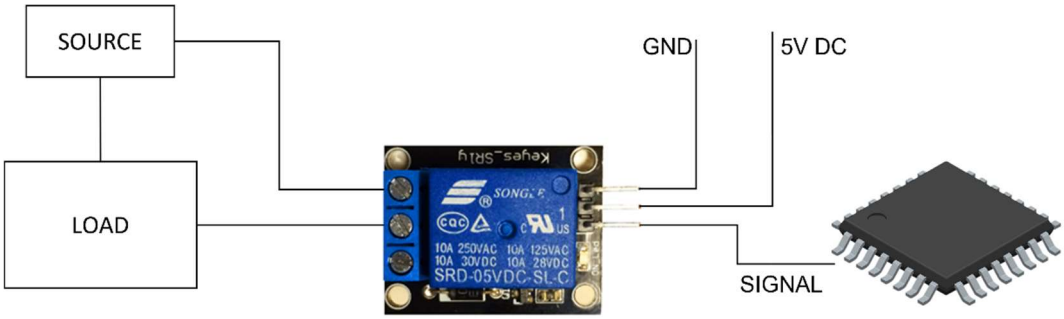

Figure. 15: Connection circuit diagram for Relay with microcontroller

CHAPTER 3: HARDWARE DESIGN

3.1. Technical requirements

With the requirement of designing a bedroom with buttons, we will be able to turn off the lights and fans through the push button, and at the same time, we measure the temperature through the DS18B20 sensor and display the temperature through the LCD. When we turn on Auto mode via the push button, we have an AI model with auto temperature output. It will compare the actual temperature with the model output temperature so that the fan can be adjusted on and off accordingly.

When the temperature from the sensor is greater than the train data at the time of input to the AI model, it will turn on the fan.

3.2. Hardware design methodology

Based on the requirements of the topic, we will choose the following hardware:

Choose a microcontroller as KIT Raspberry Pi because it is possible to train AI with a simple operating system interface and many IO ports to be able to connect to the DS18B20 sensor, relay, led, push button, and fan.

Modules connected to the Raspberry Pi Kit include:

Ds18b20 temperature sensor.

– Unlike the lm35 temperature sensor, we need an analog in the microcontroller, this one uses a 1-Wire signal transmission mechanism. With this mechanism, we can read multiple DS18B20 at the same time on the same wire.

Outstanding features:

- Using a data pin with a 64bit serial code allows us to use multiple children on the same digital pin of the microcontroller (1-Wire mechanism).
- Can supply power from 3 - 5.5V.
- The measurement from -55°C to +125°C error ±0.5°C if between -10°C to +85°C.

Relay

- Share the Raspberry Pi's 5V source with the 5VDC Fan. When there is a trigger signal to the relay, the relay will close and disconnect the Com port with the NO port.
- The power supply for the Kit will be from a backup charger with an output current to ensure the Kit works stably.

3.3. Block diagram of system

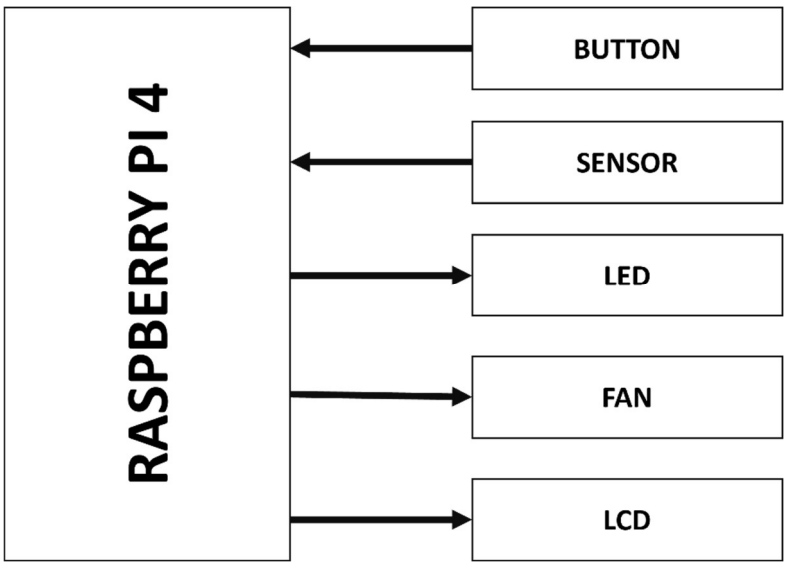

Figure. 16: Block diagram of smart bedroom.

Raspberry PI 4: we liken this to block source and block control simultaneously. It will receive a signal from the button and sensor. Raspberry Pi will receive a signal from the button to control the LED and FAN block and the signal sensor will be displayed by raspberry on the LCD screen. Next will be the Auto button to turn on AI mode, raspberry trained AI and showing the fit on the LCD to turn on the Auto Fan.

Button: when the button is pressed, it will transmit a signal to the Raspberry for control.

Sensor: here is see a blocked sensor that will transfer the signal to raspberry, the signal that is also the main signal to Raspberry to display the heat.

LCD: Block display, this is the temperature display and if auto mode is on, it will show the matching temperature pattern.

LED: This is block led, then click the button, the signal from the matching raspberry, bright led.

FAN: block Fan when the button is pressed, there is a signal from the raspberry to turn off the relay for the FAN to run. Another face, when on the auto mode, FAN will turn on and off automatically based on the temperature.

3.4. Conection circuit diagram of system

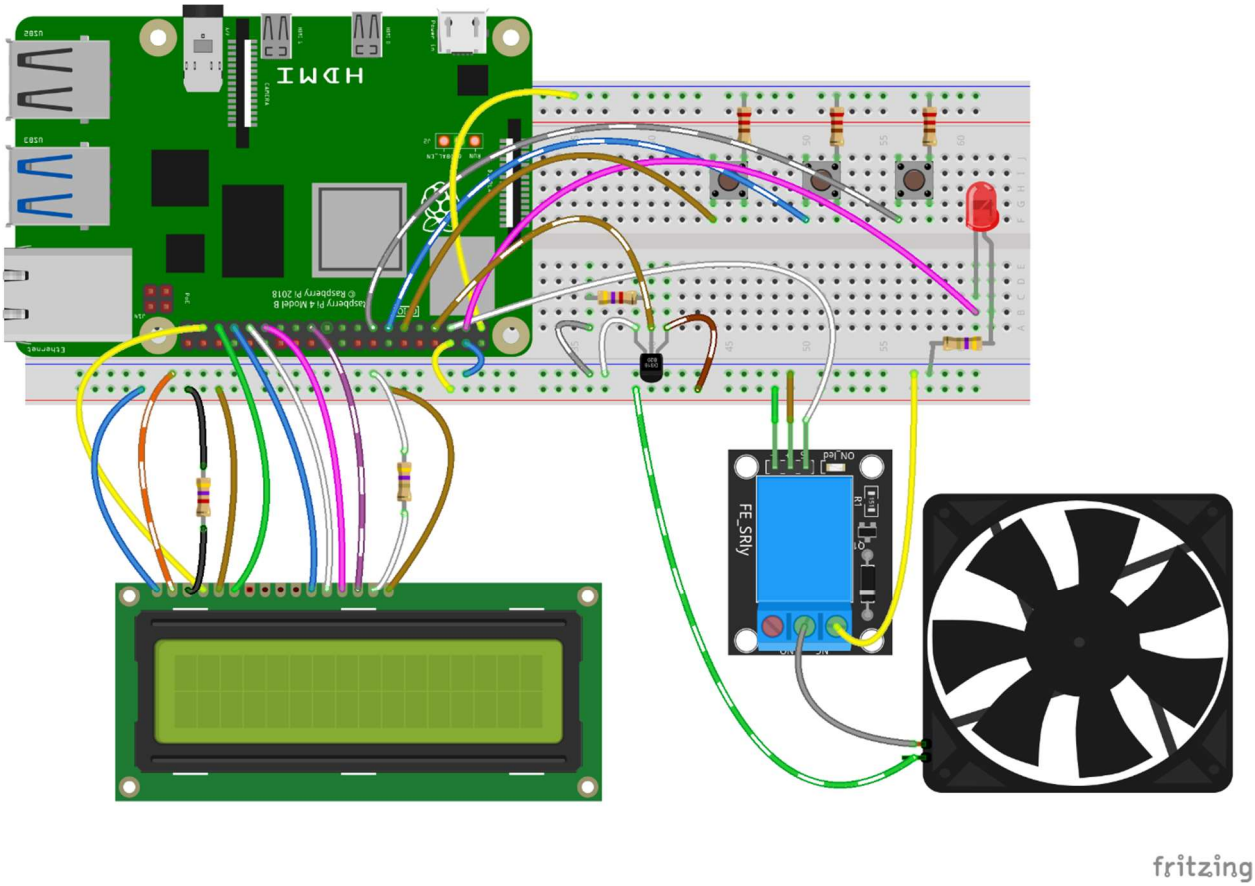

Figure. 17 : Connection circuit diagram of smart bedroom.

On the Raspberry Pi Kit, there are many GPIO pins (as shown in the picture) for us to choose to communicate with sensors, LDC, and devices.

Connect 3 buttons including LAMP, FAN, and AUTO:

– LAMP connection to GPIO17 (Pin 11).

– FAN connection to GPIO 27 (Pin 13).

– Connect to GPIO 22 (Pin15).

Connect to LED connected to GPIO 3 (Pin 5).

Relay GPIO connect 2 (Pin 3).

Connect to DS18B20 temperature sensor via GPIO 4 (Pin 7).

Connect to LCD RS, E, D4-D7 through GPIO 26 (Pin 37), GPIO 19 (Pin35), GPIO 13 (Pin 33), GPIO 6 (Pin 31), GPIO 5 (Pin 29)), GPIO 11 (Pin 23).

CHAPTER 4: SOFTWARE DESIGN

4.1. Installing the operating system for Raspberry Pi

The operating system chosen to use is Raspberry Pi OS (formerly known as Raspbian) is the introduction for the operating system for normal use on a Raspberry Pi

The operating system supports interfaces, good network communication, and good support for languages in another topic.

A SD card writing tool that works on Mac Operating system, Ubuntu, and Windows called Raspberry Pi Imager.

Hardware preparation:

- Raspberry Pi 4 Model B.
- 5V DC uses a USB-C connector with a minimum of 3A.
- A 8 GB microSD card.
- Laptop or Desktop and the card reader.
- The Ethernet wire.

Step 1: Access the link to download Raspberry Pi Imager :

Step 2: Open Raspberry Pi Imager

Step 3: Get the SD card on the card reader and connect this to the computer.

Step 4: Click “CHOOSE OS” to select OS for Raspberry Pi.

In this project, we use OS “ Raspberry Pi OS (64bit)”.

Step 5: Choose storage.

Step 6: Adjust another option.

Step 7: Choose “write” to install.

4.2. Programming for microcontroller

4.2.1. Button

Set up listeners that receive the event as the button's falling signal, and the corresponding action mechanisms used in the Actuator.

4.2.2. Actuator

The program includes the main operational definitions of the application.

Feature toggle states are used to determine the current state of the feature to use when the Button is called.

Define on-off feature used when called by Button or automatic mechanism in the program

The get state function returns the current value of the required state.

4.2.3. LCD

We will be using a Python library which is called RPLCD to drive the LCD. In detail, The RPLCD library can be installed from the Python Package Index, or PIP. PIP might already be installed on your Raspberry Pi.

Defines two common functions to use for thermal printing to LCD. Function print_temp() to display current temperature, and function print_auto() use for auto mode enabled.

4.2.4. Sensor

Enable the One-Wire interface before the Raspberry Pi can receive data from the sensor. In addition , Python program will have output temperature measurement results in degrees Celsius.

1. At the command prompt, when you enter sudo nano /boot/config.txt, then add this to the bottom of the file with : dtoverlay=w1-gpio.

2. When Exit Nano, and do reboot the Pi with sudo reboot.

3. We Log in to the Pi again, and at the command prompt we enter statment sudo modprobe w1-gpio.

4. Then we enter this statement sudo modprobe w1-therm.

Data obtained during 1-wire reception will be recorded as txt file here we use python program to get the required temperature value.

4.2.5. Survey the dataset used to train the model

This dataset contains temperature readings from IoT devices installed inside of an anonymous room (say - the administration room). The device is in alpha testing. So it was uninstalled or disabled multiple times during the entire reading period (July 28, 2018, to December 8, 2018). These random interval records and some false readings (outliers) make it more difficult to perform analysis on this data.

To demonstrate how complicated even a simple system with clear causal factors can be, I'm releasing data about the temperature in the rooms of a real house. We know with data is found in temperatures.csv, which contains three columns:

– Room: The room knows that the temperature was recorded.

– Time: In terms of the time in YMD – HMS format, the temperature was recorded.

– Temperature: The temperature in Fahrenheit.

Room	Time		Temperature
Room 1	2018-09-17	00:00:00	79.11
Room 1	2018-09-17	00:01:00	79.11
Room 1	2018-09-17	00:02:00	79.11
Room 1	2018-09-17	00:03:00	79.07

Figure. 18: Dataset in temperature.csv file

4.2.6. Artifical Intelligence Model

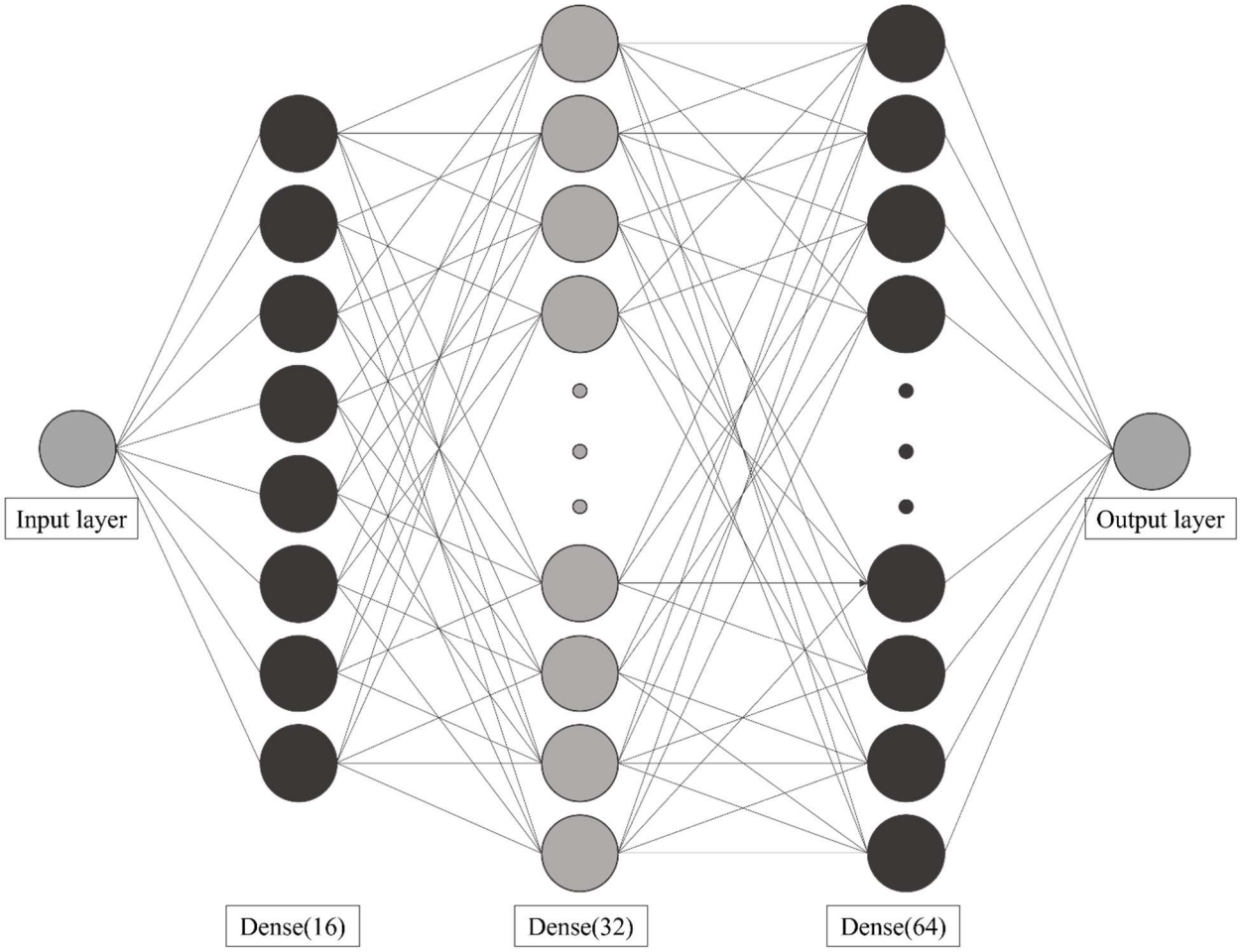

Figure. 19: Model neutral network of artifical intellegent.

Using tensorflow define a 3 layer neural network using a csv file with big data to train the model. it consists of 3 parts which are hours, minutes and room temperature respectively.

Train model with epoch number of 20, batch_size of 500 then save model as hdf5 file. This model is used to give a suitable predicted room temperature.

4.2.67. Main

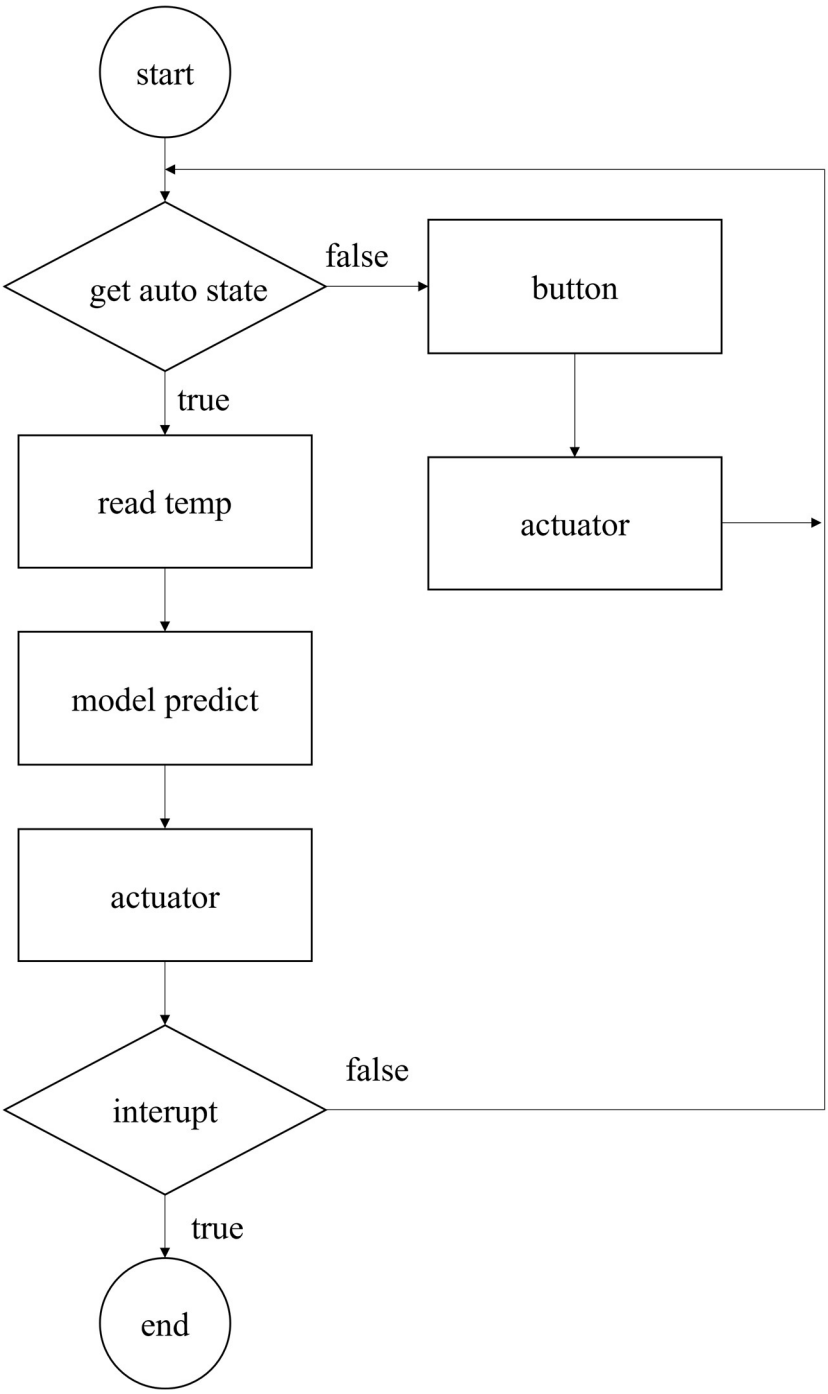

Figure. 20: Flow Chart of Main code.

Starts with non auto mode and is activated by push button to change auto state.

When in auto mode, the program will read the room temperature and return the Celsius value, then use the model file to give the appropriate temperature for the current hour and minute read by the function in the datatime library.

The actuator will allow the fan to start or stop based on the current temperature versus the appropriate temperature.

4.3. Compiler and build program in Raspberry Pi

The simple method to run a program on your Raspberry Pi at startup is to modify the .bashrc file. With the .bashrc method, your python program will run when you log in (this happens automatically

When you start up and go directly to the desktop) and also every time a new terminal is opened or when a new SSH connection is made. Put your command at the end of '/home/pi/.bashrc'.

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

echo Running at boot
python /home/pi/main.py
```

Figure. 21: Compiler and running code when startup.

CHAPTER 5: ACTUAL RESULTS

5.1. Actual results statistics

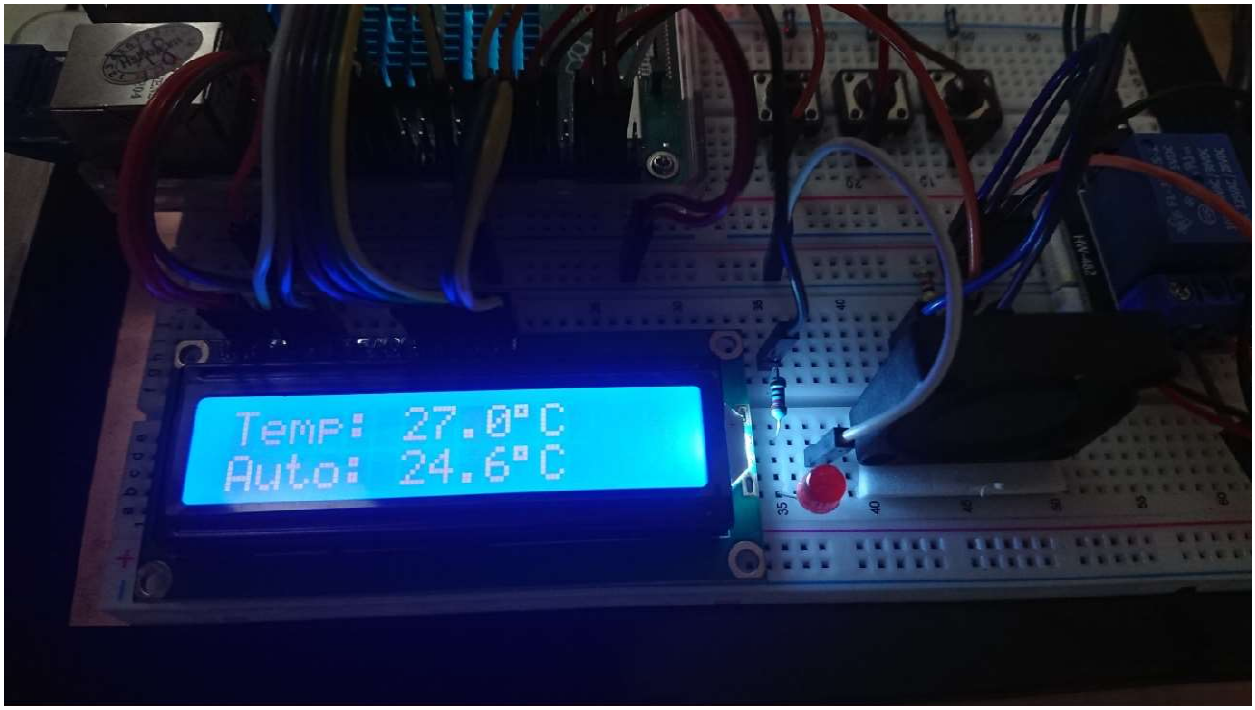

Figure. 22: Real model and LCD temperature display.

Order	Time	Temperature	Suitable Temperature	Fan
1	0: 00	26.1	24.5	On
2	3: 00	24.4	24.4	Off
3	7: 25	27.5	24.5	On
4	9: 00	27.0	24.5	On
5	14: 30	26.7	24.6	On
6	16: 00	28.6	24.4	On
7	19: 45	30.9	24.6	On
8	22: 45	26.8	24.5	On

Table. 3: Statistical table of experimental results.

Students perform the project 8 times in a changing temperature environment, the results are as follows:

The first time with the time from midnight to 0:00, the measured temperature was 26.1, the suitable temperature was 24.5, and the temperature was hotter, so the Fan automatically turned on.

The second time with the time from midnight to 3:00, the measured temperature is 24.4, the suitable temperature is 24.4, and the Fan automatically turns off.

The next time at different times, the measured temperature is higher than the suitable temperature, the Fan will automatically turn on to cool the air conditioner again.

Press the button and release			
Order	Lamp button	Fan button	Auto button
1	Fail	Fail	Fail
2	Good	Good	Good
Press and hold for a period of 2 second			
3	Good	Good	Good
Double tap			
4	Good	Good	Good

Table. 4 : Statistical table of experimental results of Button.

Along with that, we have tried the push button works well

With the condition of Press button and release:

- The first time didn't work
- The second time works fine

Provided that press and hold for a period of 2 seconds, the push button works fine

With the double tap condition, the push button works fine

5.2. Research results

The model has implemented the proposed features including intuitively controlling the device with auto and non-auto modes and easily changing between the two with the push of a button.

For non-auto mode, the buttons have well recognized almost all operations during use, including side actions such as holding the button or pressing continuously.

For auto mode, the temperature signal recognition is continuous in parallel with the adaptive temperature trained from the artificial intelligence model to adjust the fan to

operate according to the end user needs. In auto mode, the button still keeps the Listen mechanism for auto state switching and led on and off, but it won't be effective for fan control.

The disadvantage of the experiment is that the delay in execution is still high, which is caused by two main parts: the sequential programming method increases the execution time of the instruction and the devices that use the delay time are quite high. in signal conversion such as Temperature sensor and LCD overview increases system delayed.

CHAPTER 6: CONCLUTION AND DEVELOPMENT DIRECTION

6.1. Conclusion

After nearly six weeks of researching and researching professional documents, English documents, researching documents on the Internet, and the dedicated help of instructor Truong Ngoc Son, students have achieved results. as required but there are some errors.

6.2. Development direction of the topic

After the topic is done, students want to see this as basic knowledge so that they can have a passion for Embedded Programming technology, AI. The following students can develop the topic in a direction wider and deeper:

- Continue to develop applications from basic to advanced Raspberry Pi Board.
- Develop AI with better training as well as Test.
- Make the model more widely applied in life.
- Develop a more compact and optimized design.
- There are good climate and temperature conditions to test the model more
- Using model train AI by other methods.

REFERRENCE

[1] RASPBERRYPI VIỆT NAM. “Hướng dẫn cài hệ điều hành cho Raspberry Pi”. Website: raspberrypi.vn

[2] MATT . “Install RPi.GPIO Python Library”. Website: raspberrypi-spy.co.uk

[3] Ben Nuttall , Dave Jones. “gpiozero”. Website: github.com

[4] Jeff Tranter. “Control Raspberry Pi GPIO Pins from Python”. Website: ics.com/

[5] Goodfellow, I., Bengio, Y., Courville, A.: ‘ Deep learning’ (MIT Press, Boston, USA, 2016, 1st Edn).

[6] Nikhilaggarwal. “Python datetime module”. Website: geeksforgeeks.org

[7] Ben Croston. “A module to control Raspberry Pi GPIO channels”. Website: pypi.org

APPENDIX

1. Button program's executable code

```
import RPi.GPIO as GPIO
from actuator import *

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

# LAMP_BUTTON_PIN 11
# FAN_BUTTON_PIN 13
# AUTO_BUTTON_PIN 15

GPIO.setup(11, GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(13, GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(15, GPIO.IN, GPIO.PUD_DOWN)

def lamp_callback(channel):
    lamp_act()
    GPIO.remove_event_detect(11)
    lamp_detect()

def fan_callback(channel):
    fan_act()
    GPIO.remove_event_detect(13)
    fan_detect()

def auto_callback(channel):
    auto_act()
    GPIO.remove_event_detect(15)
    auto_detect()

def lamp_detect():
    GPIO.add_event_detect(11, GPIO.FALLING, callback =
lamp_callback)

def fan_detect():
    GPIO.add_event_detect(13, GPIO.FALLING, callback =
fan_callback)

def auto_detect():
    GPIO.add_event_detect(15, GPIO.FALLING, callback =
auto_callback)

lamp_detect()
fan_detect()
auto_detect()
```

2. Actuator program's executable code

```
import RPi.GPIO as GPIO
from time import sleep
from therm import *
from lcd import *

#LAMP_PIN 5
#FAN_PIN 3

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering

GPIO.setup(3, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)

lamp_state = 0
fan_state = 0
auto_state = 0

def lamp_on():
    global lamp_state
    lamp_state = 1
    GPIO.output(5, GPIO.LOW)

def lamp_off():
    global lamp_state
    lamp_state = 0
    GPIO.output(5, GPIO.HIGH)

def fan_on():
    global fan_state
    GPIO.output(3, GPIO.LOW)
    fan_state = 1
def fan_off():
    global fan_state
    fan_state = 0
    GPIO.output(3, GPIO.HIGH)

def lamp_act():
    global lamp_state
    if not lamp_state:
        lamp_on()
    else:
        lamp_off()

def fan_act():
    global fan_state
    if not fan_state:
```

```
        fan_on()
    else:
        fan_off()

def auto_act():
    global auto_state
    if auto_state:
        auto_state = 0
    else:
        auto_state = 1

def get_auto_state():
    global auto_state
    return auto_state
```

3. LCD program's executable code

```
import os
import glob
import time
from RPLCD import CharLCD
import RPi.GPIO as GPIO

lcd = CharLCD(numbering_mode=GPIO.BOARD, cols=16, rows=2,
pin_rs=37, pin_e=35, pins_data=[33, 31, 29, 23])

def temp_print(temp):
    lcd.cursor_pos = (0, 0)
    lcd.write_string("Temp: " + str(temp) + chr(223) + "C")

def auto_print(temp):
    lcd.cursor_pos = (1, 0)
    lcd.write_string("Auto: " + str(temp) + chr(223) + "C")
```

4. Sensor program's executable code

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()

    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')

    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c
```

5. Train model program's executable code

```
import tensorflow
import pandas as pd
from tensorflow import keras
from keras import Sequential
from keras.layers import Activation, Dense

model = Sequential()
model.add(Dense(16, input_dim = 1, activation = "relu"))
model.add(Dense(32, activation = "relu"))
model.add(Dense(64, activation = "relu"))
model.add(Dense(1))
model.summary()

model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['accuracy'])

def time_format(hour, minute):
    now = hour + minute/60
    return now

df = pd.read_csv("dataset.csv", on_bad_lines='skip')

hour = df.hour
minute = df.minute
temperature = df.temp

hour.to_numpy()
minute.to_numpy()
temperature.to_numpy()

time = []
temp = []
for i in range(259193):
    time.append(time_format(hour[i], minute[i]))
    temp.append(temperature[i])

model.fit(time, temp, epochs = 20, batch_size = 500)
model.save("model.hdf5")
```

6. Main program's executable code

```
from time import sleep
from therm import *
from lcd import *
from button import *
from actuator import *
import tensorflow as tf
import datetime

model = tf.keras.models.load_model("model.hdf5")

def time_now():
    now = datetime.datetime.now()
    return now.hour + now.minute/60

while True:
    while not get_auto_state():
        temp = read_temp()
        lcd.clear()
        temp_print(temp)
    while get_auto_state():
        temp = read_temp()
        auto_temp = model.predict([time_now()])
        lcd.clear()
        temp_print(temp)
        auto_print(auto_temp[0][0])
        if auto_temp < temp :
            fan_on()
        else:
            fan_off()
```