SRI KRISHNA COLLEGE OFENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

AFFLIATED TO ANNA UNIVERSITY CHENNAI

ACCREDITED BY NAAC WITH "A" GRADE

MARCH 2019

PET FEEDER

MINI PROJECT REPORT

SUBMITTED BY

ASHWIN M (18EUEC501)

NAREN S (18EUEC510)

In partial fulfillment of the requirements

for the award of the degree

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OFENGINEERING AND TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
AFFLIATED TO ANNA UNIVERSITY CHENNAI
ACCREDITED BY NAAC WITH "A" GRADE

BONAFIDE CERTIFICATE

Certified that this project report "**PET FEEDER**" is bonafide work of " **ASHWIN M**(18EUEC501), **NAREN S**(18EUEC510)" who carried out the project work under my supervision.

SIGNATURE
Dr. S.SOPHIA, Ph.D.,
Professor
Head Of The Department
Department of Electronics And Communication Engineering
Sri Krishna College of Engineering and Technology
Kuniamuthur
Coimbatore – 641008

SIGNATURE
Mr. J.R.DINESH KUMAR, M.E.,
Assistant Professor
Supervisor
Department of Electronics And Communication Engineering
Sri Krishna College of Engineering and Technology
Kuniamuthur
Coimbatore - 641008

This project report submitted for the Autonomous Mini Project Viva-voce examination held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Pets need special treatment and special care. Due to nowadays busy life style, this task is not as simple as it used to be. The goal of this work is to introduce, design and implement a smart pet system. The interaction between human and physical devices and devices in the real world is gaining more attention, and re-quires a natural and intuitive methodology to employ. According to this idea and living well, life has been a growing demand. Thus, how to raise pets in an easy way has been the main issue recently. This study examines the ability of computation, communication, and control technologies to improve human interaction with pets by the technology of the Internet of Things. This work addresses the improvement through the pet application of the ability of location-awareness, and to help pet owners raise their pet on the activity and eating control easily. Our study not only presents the key improvement of the pet feeding system involved in the ideas of the Internet of Things, but also meets the demands of pet owners, who are out for works without any trouble. The objective is to allow pet owners to automate simple things, like feeding controls. Implementing smart pet houses will assure pets owners an increased comfort and peace of mind especially when pets are unattended.

Keywords : Internet of Things, Pet feeder, Smart pet system.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Our project is automatic pet feeding system using Internet of Things. The emphasis on choosing this as the title is because, to initially give solution to a problem faced by almost everyone. Human interference on the part of taking care of pet when they are busy is difficult. And hence our system will be efficient enough to overcome the hurdles faced by human in taking care of pet. This Pet feeding System is a complete equipment for feeding the pet. Furthermore, the project is subdivided into several modules each of which has the Google assistant unique feature. They are pet food feeder and voice command.

1.1 PET FEEDER FEATURES

IoT Pet Feeder machine which is simple, efficient and cost effective. Using this machine can feed your Pet by Google Assistant from anywhere in the world. User just have to say "*Okay Google. Feed my pet*" and rest of the things will be done by this machine. User can also set a specific time using Google Assistant to feed your Pet. For example, Say "Okay Google. Feed my pet Today Morning" and it will feed your pet at previously specified time. Like this you can also set a specific time for noon, evening.

1.1.1 GOOGLE ASSISTANT FEATURES

Google Assistant offers voice commands, voice searching, and voice-activated device control, letting you complete a number of tasks after you've said the "OK Google" or "Hey, Google" wake words. It is designed to give you conversational interactions.

Google Assistant will:

- Control your devices and your smart home
- Access information from your calendars and other personal information

- Find information online, from restaurant bookings to directions, weather and news
- Control your music
- Play content on your Chromecast or other compatible devices
- Run timers and reminders
- Make appointments and send messages
- Open apps on your phone
- Read your notifications to you
- Real-time spoken translations

1.2 USAGE OF PET FEEDER

Automatic feeders are designed to dispense specific portions of dog or cat food at certain times, so they ensure that your furry friend will be well fed on time.
In addition, it has a great advantage for pet owners, as it takes care of feeding them while you are traveling. In fact, they are programmable to give meals whenever you want.
In turn, they help you control the amount of food, since it will only dispense specific amounts of food. This way you will not have to worry about your animals being overweight if they tend to eat too much.
Another advantage is that it helps you adhere to a schedule, since animals get used to a routine. Remember that your furry friends should follow them, even if your work schedules are complicated and you are very busy. With a good food dispenser, you make sure your animals feed when they touch them.

1.3 EMBEDDED SYSTEM

Embedded system do a very specific task, they cannot be programmed to do things. Embedded systems have very limited resources, particularly the memory. Generally they do not have secondary storage devices such as CDROM or floppy disk. Embedded systems have to work against some deadlines. A specific job has to be completed within a specific

time. In some embedded system, called real time systems, the deadlines are string end. Missing deadlines may cause a catastrophe –loss of life or damage to the property. Embedded system are constrained for power .As many embedded system operate through a battery, the power consumption has to be very low. Some embedded systems have to operate in extreme environment such as very high temperature and humidity. Embedded systems that address the consumer market is very cost sensitive. Unlike desktop computers in which the hardware platform is dominated by intel and the operating system is dominated by microsoft, there is a wide variety of processor and the operating systems for embedded system. So choosing the right platform is the complex task.

1.4 NODE MCU

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated kits. Both the firmware and prototyping board designs are source. The firmware uses the Lau scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects).

1.5 SERVO MOTOR

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor, although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

CHAPTER 2

PROPOSED SYSTEM

2.1 PROPOSED TECHNIQUE

Aim of this Pet Feeder is to continuously takes input data from the Google Assistant and stores the on and off values in IO-Adafruit. if we have to feed the pet, we need to send voice command to Google Assistant, by which it gets activated. It also gets activated by allocated time period which is defined by user. Once the voice command is received and Servo Motor gets activated. The Node MCU receives the real-time from wifi and user defined time from voice command. The feeding process is done by comparing the real-time and user defined time in node MCU. Once the comparison is done servo motor is activated.

2.2 ADVANTAGES OF PROPOSED SYSTEM

- Can feed pets at desired time
- Controlled through Voice command
- Small in size
- Less cost

2.3 BLOCK DIAGRAM

This is the block diagram of pet feeder. This shows the overall view of the pet feeder circuit. The blocks connected here are LCD display, I2C module, Google assistant, Servo motor, Power supply.

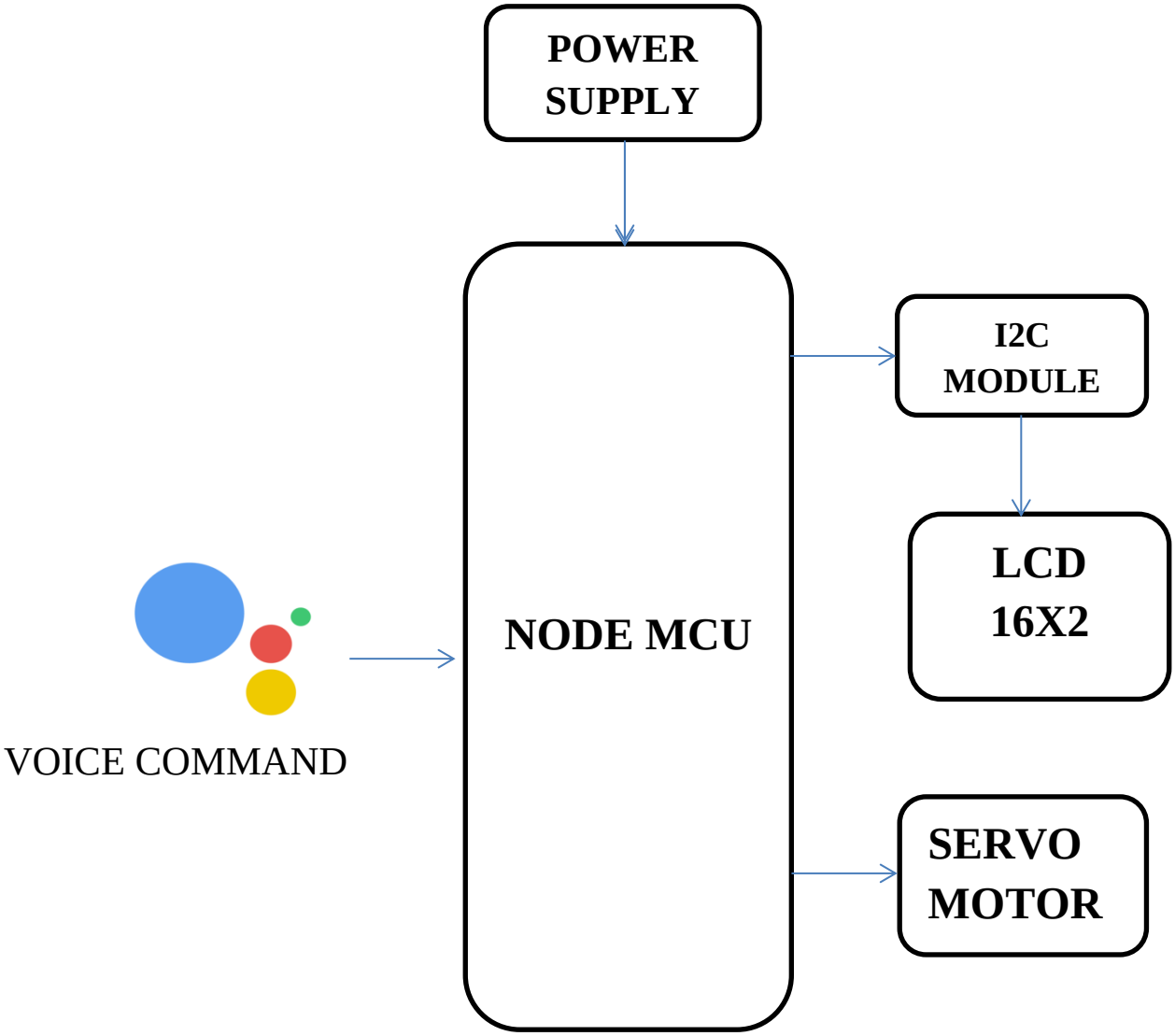


Figure 2.1 : The overall block diagram

CHAPTER 3

WORKING OF THE SYSTEM

3.1 CONCEPT AND OVERVIEW

There are many ways to implement a pet feeder: you can set it to fill up the bowl at a certain time, you can command it to fill up whenever it gets empty, or maybe to give your dog food after they follow a set of orders that you taught them.

In this specific project, the command instruction is given by the user through voice command by the platform google.

In addition, we also decided to add the option for user defined to feed the pet at specific time interval. The combination of MQTT and IFTTT is used to integrated for a seamless connection between user and pet feeder. After the instruction the received the servo motor is activated.

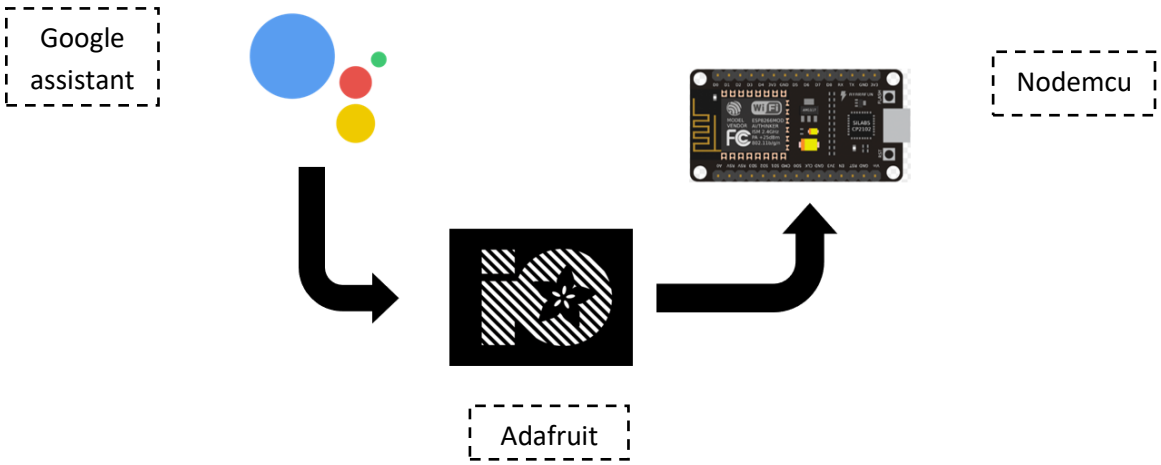


Figure 3.1 Overview of the system

3.2 MQTT INTEGRATION

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

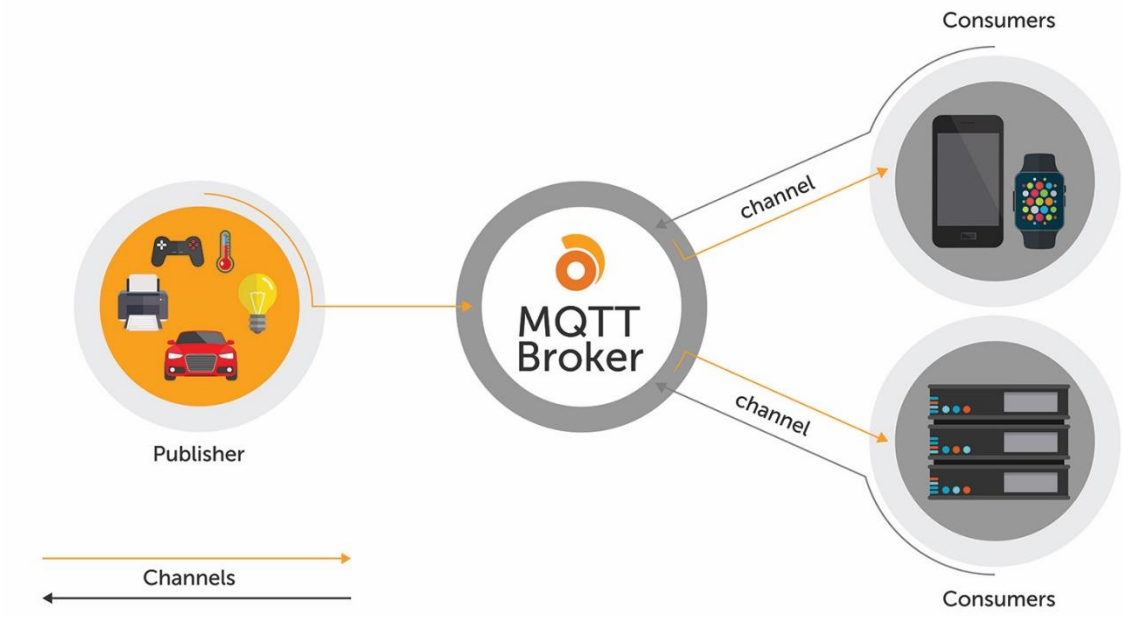


Figure 3.2 MQTT protocol

3.3 ADAFRUIT.IO CONFIGURATION

Adafruit.io is a *cloud service* - that just means we run it for you and you don't have to manage it. You can connect to it over the Internet.

- Display your data in real-time, online
- Make your project internet-connected: Control motors, read sensor data, and more!
- Connect projects to web services like Twitter, RSS feeds, weather services, etc.
- Connect your project to other internet-enabled devices
- The best part? All of the above is do-able for free with Adafruit IO

3.4 IFTTT CONFIGURATION

IFTTT derives its name from the programming conditional statement “if this, then that.” What the company provides is a software platform that connects apps, devices and services from different developers in order to trigger one or more automations involving those apps, devices and services.

Here are just three *if this, then that* automations you can run with IFTTT:

* *If* you make a call on your Android phone, *then* a log of that call is added to a Google spreadsheet.

* *If* you add a new task to your Amazon Alexa to-dos, *then* it will be added to your iOS Reminders app.

* *If* the International Space Station passes over your house, *then* you’ll get a smartphone notification about it. (Yes, this is an actual IFTTT applet.)

CHAPTER 4
HARDWARE DESCRIPTION

4.1 NODE MCU

4.1.1 Features:

NodeMCU is an open source IoT platform. This includes firmware which runs on the ESP8266 Wi-Fi Module from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the dev kits. NodeMCU firmware was developed so that AT commands can be replaced with Lua scripting making the life of developers easier. So it would be redundant to use AT commands again in NodeMCU.

The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif.

4.1.1.1 Technical specifications

Developer : ESP8266 Open source Community
Type : Single-board microcontroller
Operating system : XTOS
CPU : ESP8266
Memory : 128kBytes
Storage : 4MBytes
Power By : USB
Power Voltage : 3v ,5v (used with 3.3v Regulator which inbuilt on Board using Pin VIN)
Code : Arduino Cpp

IDE Used : Arduino IDE

GPIO : 10

4.1.1.2Power supply for the NODEMCU:

NodeMCU is an open source IoT platform including a firmware which runs on the ESP8266 with the Espressif Non-OS SDK, and hardware based on the ESP-12 module. The device features 4MB of flash memory, 80MHz of system clock, around 50k of usable RAM and an on chip Wifi Transceiver.

Power to the NodeMCU v2 is supplied via the on-board USB Micro B connector or directly via the “VIN” pin. The power source is selected automatically.

The device can operate on an external supply of 6 to 20 volts. If using more than 12V, the voltage regulator may overheat and damage the device. The recommended range is 7 to 12 volts.

.

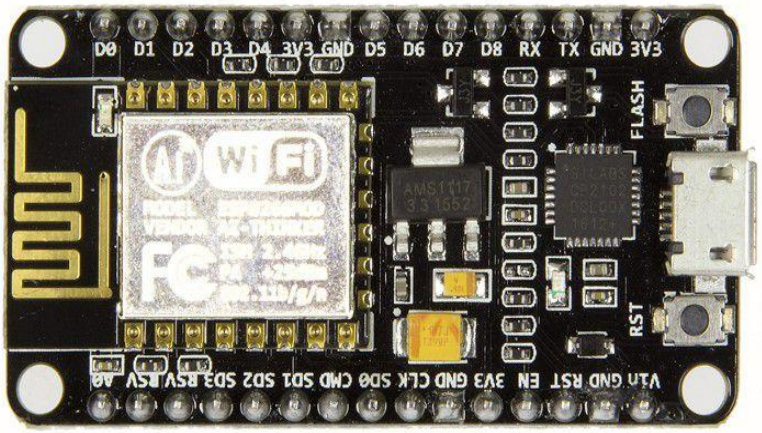


Figure 4.1 : Overview of the NODEMCU

4.1.2 PIN DETAILS

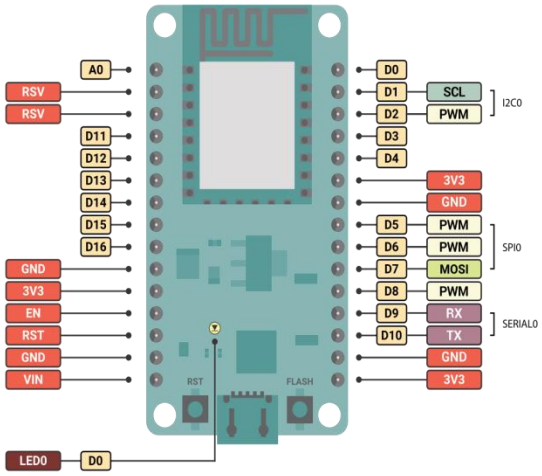


Figure 4.2 : Clearer view of the NodeMCU

4.1.2.1 Power pins

VIN : The input voltage to the NodeMCU board when it's using an external power source (as opposed to be 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if
supplying voltage via the power jack, access it through this pin.

5V : The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

3.3V: A 3.3 volt supply generated by the on-board regulator. Maximum current drawn is 50 mA.

GND : Ground pin

4.1.2.2 Memory

The ESP8266 has 32 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

4.1.2.3 Input and Output

Power **Pins** There are four power **pins** viz. one VIN **pin** & three 3.3V **pins**. The VIN **pin** can be used to directly supply the **ESP8266** and its peripherals, if you have a regulated 5V voltage source. The 3.3V **pins** are the **output** of an on-board voltage regulator.

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the

corresponding pins of the Atmega8U2 USB-to-TTL Serial chip .

- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21(interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is in HIGH value, the LED is on, when the pin is LOW, it's off.
- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library. Note that these pins are not in the same location as the I2C pins on the Duemilanove. The Mega328p has 6 analog inputs, each of which provide 10 bits of

resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference()function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller.

4.1.3 COMMUNICATION

The NodeMCU has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The NodeMCU provides four hardware UARTs for TTL (5V) serial communication.

An NodeMCU on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically). The NodeMCU software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the NodeMCU and USB connection to the computer (but not for serial communication on pins 0 and 1).ASoftwareSerial library allows for serial communication on any of the Mega's digital pins. The NodeMCU also supports I2C (TWI) and SPI communication.

4.1.4 PROGRAMMING

The NodeMCU can be programmed with the Arduino software The ESP8266 on the NodeMCU comes preburned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).You can also bypass the boot

loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

4.1.5 AUTOMATIC (SOFTWARE) RESET

Software reset for ESP8266 is required when you get trouble to connect WiFi router. Let's see the use of software restart. This example program will show you software reset in a loop before it reaches to its max value.

ESP. reset() is a hard reset and can leave some of the registers in the old state which can lead to problems, its more or less like the reset button on the PC.

ESP. restart() tells the SDK to reboot, so its a more clean reboot.

4.2 I2C MODULE

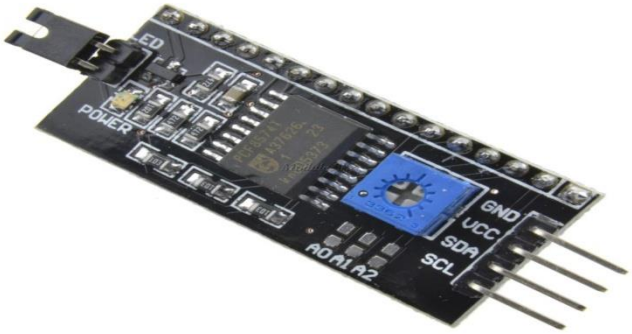


Figure 4.3 I2C module

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). Like SPI, **I2C** is synchronous, so the output of bits is

synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

Figure 4.4 Pinout diagram for I2C module

Many slave devices are interfaced to the microcontroller with the help of the I2C bus through I2C level shifter IC for transferring the information between them. The I2C protocol used to connect a maximum of 128 devices that are all connected to communicate with the SCL and SDL lines of the master unit as well as the slave devices. It supports Multimaster communication, which means two masters are used to communicate the external devices.

4.3 LCD DISPLAY

The **LCD** works on the **principle** of blocking light. While constructing the **LCD's**, a reflected mirror is arranged at the back. An electrode plane is made of indium-tin oxide which is kept on top and a polarized glass with a polarizing film is also added on the bottom of the device.

An **LCD** is an electronic **display** module which uses **liquid crystal** to produce a visible image. The 16×2 **LCD display** is a very basic module commonly used in DIYs and circuits. The 16×2 translates o a **display** 16 characters per line in 2 such lines. In this **LCD** each character is **displayed** in a 5×7 pixel matrix.

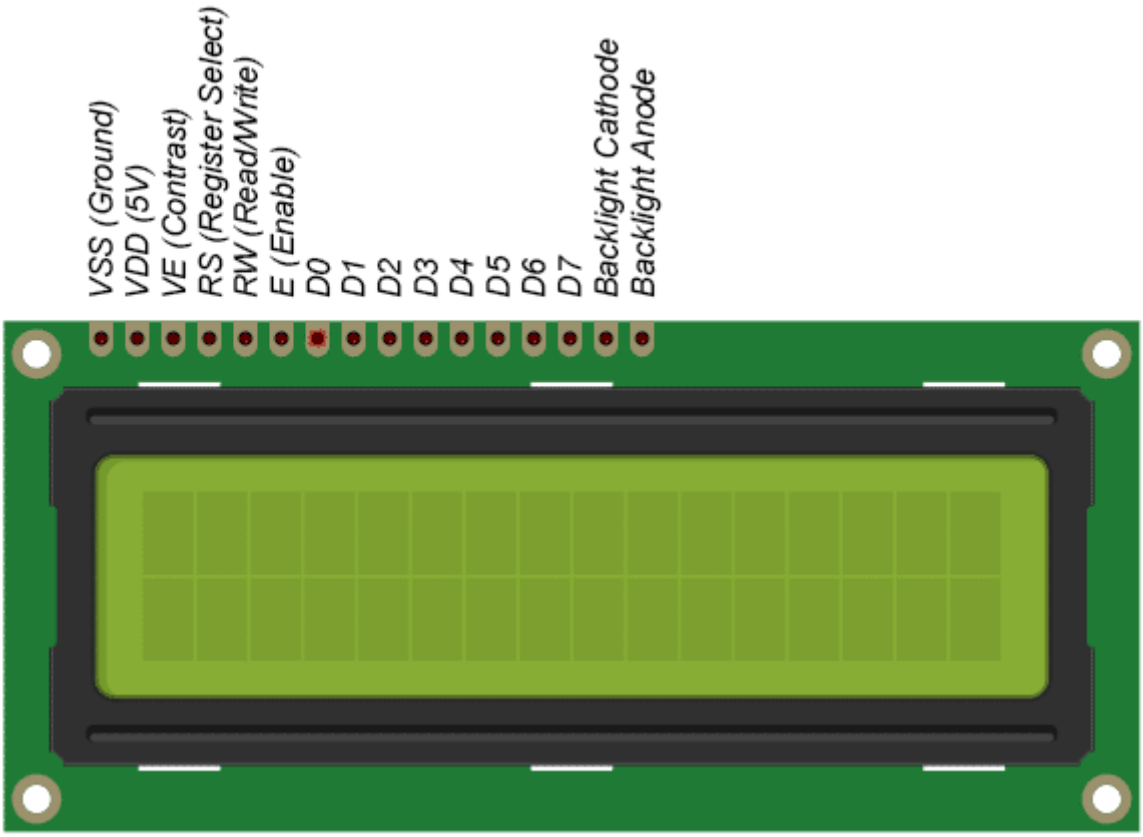


Figure 4.5 LCD 16X2 DISPLAY MODULE

4.4 SERVO MOTOR

Servo motor works on the PWM (Pulse **Width** Modulation) principle, which means its angle of rotation is controlled by the duration of pulse applied to

its **control** PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears.

Figure 4.6 Vibration sensor

Applications

- Used as actuators in many robots like Biped Robot, Hexapod, robotic arm etc..
- Commonly used for steering system in RC toys
- Robots where position control is required without feedback
- Less weight hence used in multi DOF robots like humanoid robots

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 HARDWARE REQUIREMENTS

A personal computer (PC) is any general-purpose computer whose size, capabilities, and original sales price make it useful for individuals, and which is intended to be operated directly by an end-user with no intervening computer operator. In contrast, the batch processing or time-sharing models allowed large expensive mainframe systems to be used by many people, usually at the same time. Large data processing systems require a full-time staff to operate efficiently.

A computer user will apply application software to carry out a specific task. System software supports applications and provides common services such as memory management, network connectivity, or device drivers; all of which may be used by applications but which are not directly of interest to the end user.

A simple, if imperfect analogy in the world of hardware would be the relationship of an electric light bulb (an application) to an electric power generation plant (a system). The power plant merely generates electricity, not itself of any real use until harnessed to an application like the electric light that performs a service that benefits the user.

Windows 7 includes a number of new features, such as advances in touch and handwriting recognition, support for virtual hard disks, improved performance on multi-core processors, improved boot performance, Direct Access, and kernel improvements. Windows 7 adds support for systems using multiple heterogeneous graphics cards from different vendors (Heterogeneous Multi-adapter), a new version of Windows Media Center, a Gadget for Windows Media Center, improved media features, the XPS

Essentials Pack and Windows Power Shell being included, and a redesigned Calculator with multiline capabilities including Programmer and Statistics modes.

Many new items have been added to the Control Panel, including Clear Type Text Tuner, Display Color Calibration Wizard, Gadgets, Recovery, Troubleshooting, Workspaces Center, Location and Other Sensors, Credential Manager, Biometric Devices, System Icons, and Display.

The Windows Security Center has been renamed to Windows Action Center (Windows Health Center and Windows Solution Center in earlier builds), which encompasses both security and maintenance of the computer. Ready boost on 32bit editions now supports up to 256 Gigabytes of extra allocation.

The default setting for User Account Control in Windows 7 has been criticized for allowing untrusted software to be launched with elevated privileges without a prompt by exploiting a trusted application. Microsoft's Windows kernel engineer Mark Russinovich acknowledged the problem, but noted that malware can also compromise a system when the users agree to a prompt. Windows 7 also supports images in the RAW image format through the addition of Windows Imaging Component-enabled image decoders, which enables raw image thumbnails, previewing and metadata display in Windows Explorer, plus full-size viewing and slideshows in Windows Photo Viewer and Windows Media Center.

5.1.1 Nodemcu board

NodeMCU is open source platform, their hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 Wi-Fi Module. There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.

NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols. To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement. There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement.

5.2 Software requirements

5.2.1 Arduino IDE

A program for Arduino hardware may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit AVR and 32-bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

The Arduino integrated (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two

basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avr dude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

CHAPTER 6

HARDWARE PHOTOGRAPHY

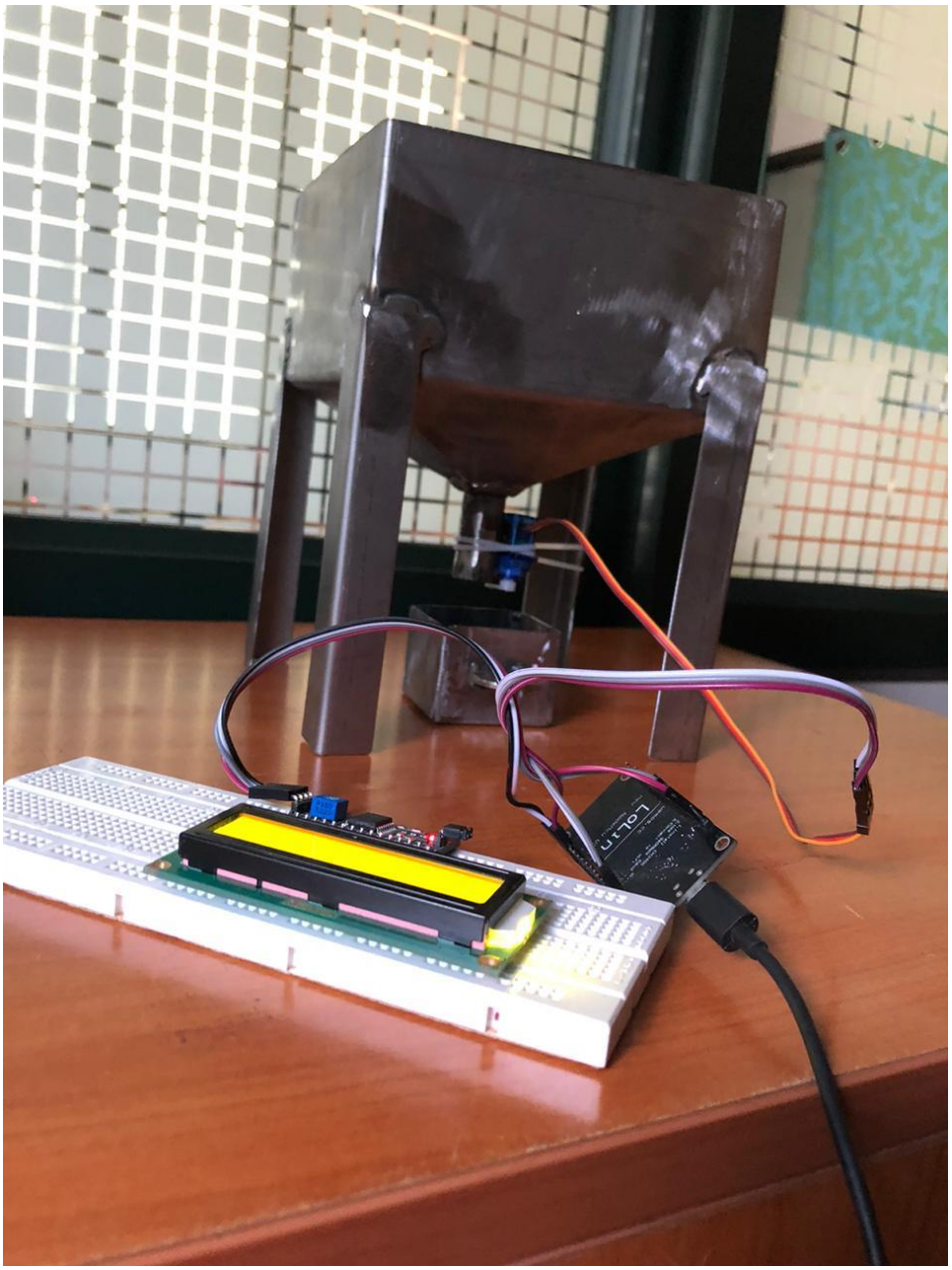


Figure 6.1 : Hardware Photography

CHAPTER 7

PROGRAM

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <Servo.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800,60000);
Servo servo;
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define WIFI_SSID "Naren"
#define WIFI_PASS "12345678990"
#define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "Narensraj"
#define MQTT_PASS "aio_vYUF15qYeIwEbxZpvrPlBG4CELgp"
int SERVO_PIN = D3;    // The pin which the servo is attached to
int CLOSE_ANGLE = 0;  // The closing angle of the servo motor arm
int OPEN_ANGLE = 70;  // The opening angle of the servo motor arm
int  hh, mm, ss;
int feed_hour = 0;
int feed_minute = 0;
```

```
//Set up MQTT and WiFi clients
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, MQTT_SERV, MQTT_PORT, MQTT_NAME,
MQTT_PASS);
//Set up the feed you're subscribing to
Adafruit_MQTT_Subscribe onoff = Adafruit_MQTT_Subscribe(&mqtt, MQTT_NAME
"/f/PET FEEDER");
boolean feed = true; // condition for alarm
void setup()
{
  Serial.begin(115200);
  timeClient.begin();
  Wire.begin(D2, D1);
  // initialize the LCD
  lcd.begin();
 // Turn on the blacklight and print a message.
  lcd.backlight();
  lcd.print("  PET FEEDER  ");
  //Connect to WiFi
  Serial.print("\n\nConnecting Wifi... ");
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
  }
```

```
  Serial.println("OK!");
 //Subscribe to the onoff feed
  mqtt.subscribe(&onoff);
  servo.attach(SERVO_PIN);
  servo.write(CLOSE_ANGLE);
  }
void loop()
{
   MQTT_connect();
   timeClient.update();
   hh = timeClient.getHours();
   mm = timeClient.getMinutes();
   ss = timeClient.getSeconds();
   lcd.setCursor(0,0);
   lcd.print("Time:");
    if(hh>12)
  {
    hh=hh-12;
    lcd.print(hh);
    lcd.print(":");
    lcd.print(mm);
    lcd.print(":");
    lcd.print(ss);
    lcd.println(" PM  ");
  }
  else
  {
```

```
  lcd.print(hh);
  lcd.print(":");
  lcd.print(mm);
  lcd.print(":");
  lcd.print(ss);
  lcd.println(" AM  ");
}
 lcd.setCursor(0,1);
 lcd.print("Feed Time:");
 lcd.print(feed_hour);
 lcd.print(':');
 lcd.print(feed_minute   );

Adafruit_MQTT_Subscribe * subscription;
while ((subscription = mqtt.readSubscription(5000)))
{

  if (subscription == &onoff)
  {
    //Print the new value to the serial monitor
    Serial.println((char*) onoff.lastread);

  if (!strcmp((char*) onoff.lastread, "ON"))
    {

      open_door();
      delay(500);
```

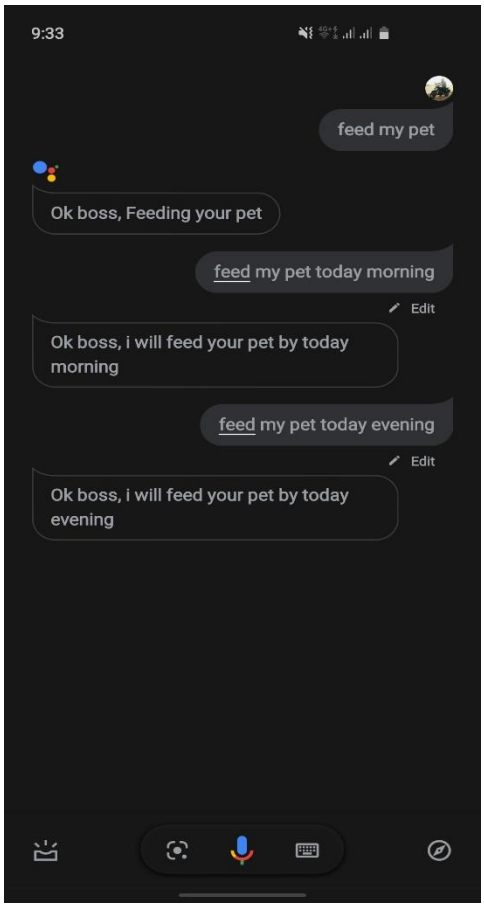
```
      close_door();
     }
    if (!strcmp((char*) onoff.lastread, "morning"))
    {
     feed_hour = 10;
     feed_minute = 30;
    }
    if (!strcmp((char*) onoff.lastread, "afternoon"))
    {
     feed_hour = 1;
     feed_minute = 30;
    }
    if (!strcmp((char*) onoff.lastread, "evening"))
    {
     feed_hour = 4;
     feed_minute = 25;
    }
    if (!strcmp((char*) onoff.lastread, "night"))
    {
     feed_hour = 10;
     feed_minute = 30;
    }
   }
  }
  if( hh == feed_hour && mm == feed_minute &&feed==true) //Comparing the current
time with the Feed time
```

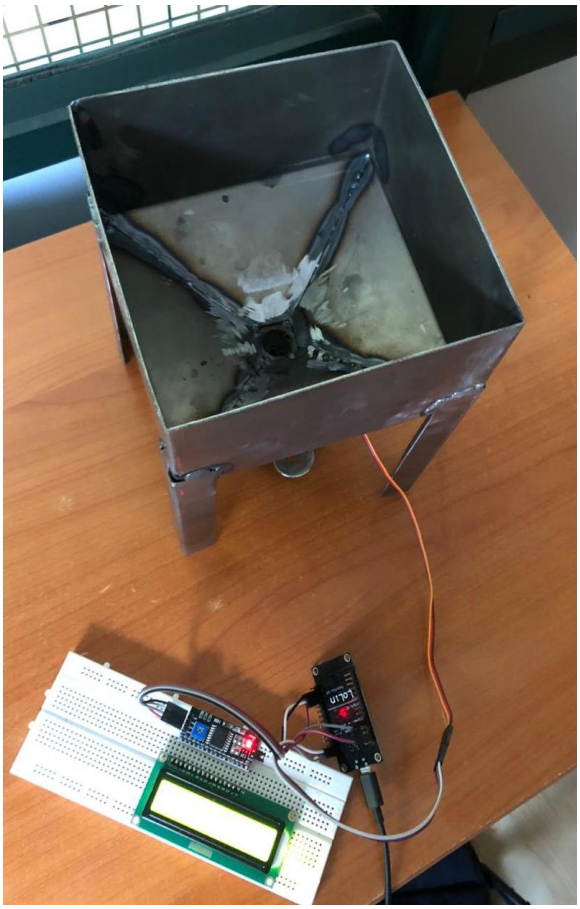
```
    {
      open_door();
      delay(500);
      close_door();
      feed= false;
      }
}

void MQTT_connect()
{
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected())
  {
    return;
  }

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) // connect will return 0 for connected
  {

      mqtt.disconnect();
      delay(5000);  // wait 5 seconds
      retries--;
      if (retries == 0)
      {
```

```
        // basically die and wait for WDT to reset me
        while (1);
      }
  }

}

void open_door(){

  servo.write(OPEN_ANGLE);   // Send the command to the servo motor to open the trap door
}

void close_door(){

  servo.write(CLOSE_ANGLE);   // Send te command to the servo motor to close the trap door
}
```

CHAPTER 8

RESULT





CHAPTER 9

CONCLUSION & FUTURE ENHANCEMENT

In the proposed project embedded system technique is incorporated for pet feeder. We have detected the latitude and longitude of the accident area and the information is sent to computer or mobile through GSM module. In the proposed project whenever accident occurs in vehicle then the device send message to given mobile device. This system shows the location of vehicle on the LCD display connected to it just to make sure the working condition of the microcontroller. The results obtained found to be more appropriate and the operator can send the nearest ambulance driver to the spot immediately in order to save life. Even though the chance of saving the person is complex there is slight chance where we can save life of a person with this project.

REFERENCE

1. https://www.researchgate.net/publication/330702107_Design_of_Pet_Feeder_using_Web_Server_as_Internet_of_Things_Application
2. https://ifttt.com/adafruit
3. https://learn.adafruit.com/welcome-to-adafruit-io
4. www.iraj.in › journal › journal_file › journal_pdf
5. https://create.arduino.cc › projecthub › circuito-io-team › iot-pet-feeder-10