Cat Feeder Code

Jonathan de Laine

```
#include <Stepper.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoOTA.h>
#include <WiFi.h>
#include <analogWrite.h>
#include "XT_DAC_Audio.h"
#include "bells.h"

 //#include <ESP8266WiFi.h>
 //#include <ESP8266WiFiMulti.h>

// wifi
//ESP8266WiFiMulti wifiMulti;
WiFiClientSecure client;

const char* ssid = "walak";
const char* password = "mateenisfat123";
//const char* ssid = "Sarooshki";
//const char* password = "12345678";

// stepper
const int stepsPerRev = 360.0/1.8;
const int motorspeed = 27;
//Stepper pins

//14 32 15 33
//Stepper myStepper(stepsPerDose, D1, D2, D3, D4);

Stepper myStepper(stepsPerRev, 14, 32, 15, 3);

//Pins 21 17

//int enA = D5;
int enA = 21;

//int enB = D6;
int enB = 17;
int motorPower = 990;

//Made this available, = 0
```

```
float percentageFood = 0;
// ultrasonic
long t;
int trigger = 16;
int echo = 19;
float mm;
float inches;

//originally 27
float max_food = 15.00;


// telegram
#define BOTtoken "983507744:AAHW0hfHL9IdAt4asLsLrYt24dWjaYU4_qI"
UniversalTelegramBot bot(BOTtoken, client);
//UniversalTelegramBot bot(BOTtoken, ssid);
int Bot_mtbs = 1000;
long Bot_lasttime;
bool Start = false;

//Music playing setup
 XT_Wav_Class JBRock(bells_wav);

 //Initialize DAC functionality for pin 25 on ESP32
 XT_DAC_Audio_Class DacAudio(25,0);

void setup()
{
  // Serial setup
  Serial.begin(115200);

  // Wifi connection setup
  /*wifiMulti.addAP("REPLACEME", "REPLACEME");
  wifiMulti.addAP("REPLACEME", "REPLACEME");
  while (wifiMulti.run() != WL_CONNECTED) {         // Wait for the Wi-Fi to connect: scan for Wi-Fi networks, and connect to the strongest of the networks above
    delay(1000);
    Serial.print('.');
  }
  Serial.print(WiFi.localIP());
*/
WiFi.disconnect();
WiFi.begin(ssid,password);

while (WiFi.status() != WL_CONNECTED)
{
  delay(1000);
  Serial.println("Connecting to WiFi...");
}
```

```

Serial.println("Connected to Wifi Network");
Serial.println(WiFi.localIP());


  // pins setup
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  //Ultrasonic Sensor Pins
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  // stepper speed

  myStepper.setSpeed(motorspeed);

  //Initialize Music Playback Settings

  //Loop setup here
  JBRock.RepeatForever=false;
  //DacAudio.FillBuffer();

  // OTA setup
  ArduinoOTA.setHostname("catFeeder");
  ArduinoOTA.begin();
}


//Defining Functions


// calc remaining food in %
void calcRemainingFood()
{
  //Sending the pulse out
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);

  //Begin reading pulse
  //t = (pulseIn(echo, HIGH) / 2);
    t = pulseIn(echo, HIGH);
  if (t == 0.00)
  {
    Serial.println("Failed to read from RCWL-1601");
    delay(1000);
    return;
  }
  //distance = float(t * 0.0343);

//Calculating distance/food remaining
 inches = microsecondsToInches(t);
 mm = microsecondsToMillimeters(t);


  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(mm);
  Serial.print("mm");
  Serial.println();
  Serial.println(t);

  //Calculate percentage of food
  //percentageFood = (100 - ((100 / max_food) * cm));
  percentageFood = (mm - 163)/(-0.23);
  if (percentageFood < 0.00)
  {
    percentageFood = 0.00;
  }
  if (percentageFood > 100.00)
  {
    percentageFood = 100;
  }
  Serial.print("Remaining food:\t");
  Serial.print(percentageFood);
  Serial.println(" %");
  delay(500);
}


// feeds cats
void feedCats()
{
  analogWrite(enA, motorPower);
  analogWrite(enB, motorPower);
  myStepper.step(-stepsPerRev);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  delay(2000);
}

// clean feeder
void cleanFeeder()
{
  analogWrite(enA, motorPower);
  analogWrite(enB, motorPower);
  myStepper.step(-3*stepsPerRev);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  delay(1000);
}

// telegram message handler
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    //Reads message from text message
    String text = bot.messages[i].text;

    String from_name = bot.messages[i].from_name;
    if (from_name == "") from_name = "Guest";
    if ( chat_id != "1052480684")
    {
      Serial.println(String(chat_id));
      bot.sendMessage(chat_id, "Only Masood or Jon gets to feed the cats!.", "");
    }
    else if ( chat_id == "1052480684")
    {
      if (text == "/dispense")
      {
        if (percentageFood == 0.00)
        {
          //DacAudio.Play(&JBRock);
          feedCats();
          bot.sendMessage(chat_id, "Cats fed. There was no food! (Ultrasonic measured distance: " + String(mm) + " mm).", "");
          calcRemainingFood();
          char buffer[5];
          bot.sendMessage(chat_id, "Current food: " + String(percentageFood) + " % (Ultrasonic measured distance: " + String(mm) + " mm).", "");
        }
        else
        {
          //DacAudio.Play(&JBRock);
          //playmusic();
          feedCats();
          bot.sendMessage(chat_id, "Cats fed! Remaining food: " + String(percentageFood) + " %. Ultrasonic measured distance: " + String(mm) + " mm.", "");
        }
      }
      if (text == "/check")
      {
        calcRemainingFood();
        char buffer[5];
        bot.sendMessage(chat_id, "Remaining food: " + String(percentageFood) + " % (Ultrasonic measured distance: " + String(mm) + " mm).", "");
      }
      if (text == "/clear")
      {
        //Changed this to cleanfeeder from feedcats
        cleanFeeder();
        char buffer[5];
        bot.sendMessage(chat_id, "Feader cleaned. Remaining food: " + String(percentageFood) + " % (Distance to food: " + String(mm) + " mm).", "");
      }
      if (text == "/ip")
      {
        String catFeederIP = WiFi.localIP().toString();
        bot.sendMessage(chat_id, "catFeeder local IP address: " + (catFeederIP), "");
      }
      if (text == "/help" || text == "/start")
      {
        String welcome = "Welcome to our awesome ESP32 catFeeder!\n";
        welcome += "/clear : Cleans the feeder by rotating several times.\n";
        welcome += "/dispense : Delivers one dose of feed.\n";
        welcome += "/help : Outputs this help message.\n";
        welcome += "/ip : Prints catFeeder local IP.\n";
        welcome += "/check : Returns remaining feed quantity.\n";
        bot.sendMessage(chat_id, welcome, "Markdown");
      }
    }
  }
}


void loop()
{
  //DacAudio.FillBuffer();
  ArduinoOTA.handle();
  calcRemainingFood();
  Serial.println(WiFi.localIP());
  if (millis() > Bot_lasttime + Bot_mtbs)
  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while (numNewMessages)
    {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }

    Bot_lasttime = millis();
  }
  delay(500);
}

long microsecondsToInches(long microseconds)
{
  // According to Parallax's datasheet for the PING))), there are 73.746
  // microseconds per inch (i.e. sound travels at 1130 feet per second).
  // This gives the distance travelled by the ping, outbound and return,
  // so we divide by 2 to get the distance of the obstacle.
  // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
  return microseconds / 74.0f / 2.0f;
}
long microsecondsToMillimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the object we
  // take half of the distance travelled.
  return microseconds / 2.90f / 2.0f;
}
void playmusic()
{
DacAudio.FillBuffer();
if(JBRock.Playing == false)
{
  DacAudio.Play(&JBRock);
  Serial.println("Playing Music");
}
}
//}
```