IoT Pond Depth Sensor

Summer 2021

Technical Report

Michael Liang (ml2226)

August 28, 2021

Contents

1 Design Requirements

1.1 Constraints

- 3D Printable and Must Enclose the Sensor and Arduino
- Must be able to easily open the sensor case

1.2 Objectives

- Use Snap-Fit Technology and/or Topology Optimization
- Have ample room for wiring

2 High Level Description

The Pond Depth Sensor utilized an Arduino Nano 33 IoT, a HR-SC04 Ultrasonic Sensor, and a custom designed and 3D printed sensor case to elevate the assembly above the pond water. The data retrieved was analyzed and sent using HTTP POST Requests using the Arduino API, Ubidots API, and IFTT to notify emergency depth levels of the pond water directly through phone messaging.

2.1 Arduino Nano 33 IoT Casing

Figure 1: Sensor Case - Bottom

The bottom sensor casing offers an attachment to the Arduino microcontroller. It offers an indent for the place where the lock mechanism clips into. On the side, it has an open section for the micro-usb cable. The bottom casing (Section 2.1) casing offers a honeycomb pattern to allow for increased airflow and reduced use of material when 3D Printing.

2.2 HC-SR04 Casing

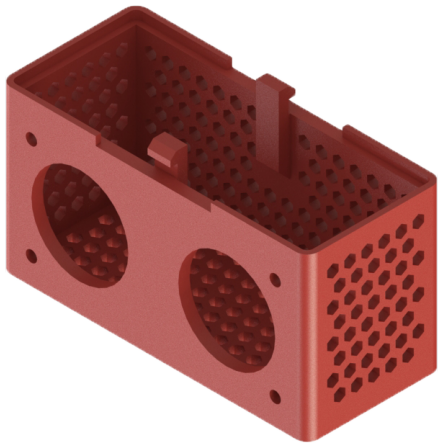

Figure 2: Sensor Case - Top

The top sensor casing offers spacious room for the HC-SR04 Ultrasonic Sensor and the wires it needs to connect to the Arduino pins. With the overhanging pins that connect to the indentation on the bottom casing of sensor box, it serves the purpose of having an easy to open lock mechanism. The top casing (Section 2.2) casing also offers a honeycomb pattern to allow for increased airflow and reduced use of material when 3D Printing.

2.3 Arduino Nano 33 IoT Modifications

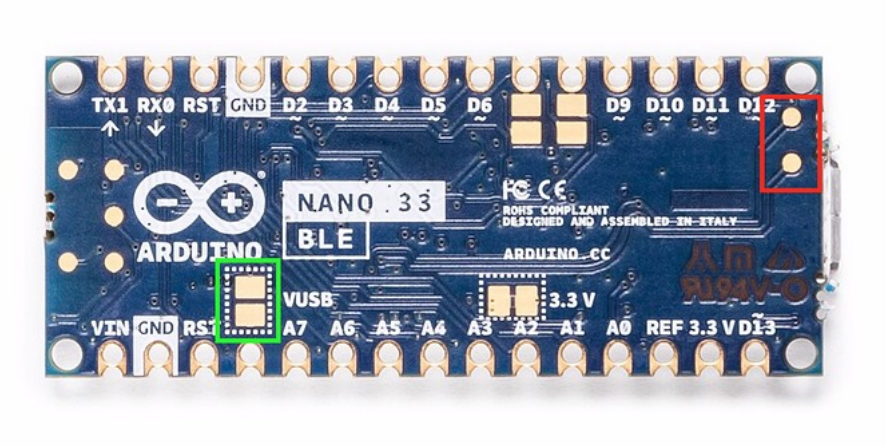

Figure 3: Solder Bridge

The Arduino Nano 33 IoT runs at a voltage of 3.3V; however, the ultrasonic sensor runs at a voltage of 5V. As a result of this, we have to create a solder bridge to activate the 5V operating voltage for the Arduino. This is done by creating a solder between the two pads marked by the VUSB or the green box (Figure 3).

2.4 Arduino Nano 33 IoT Integration Code

```
/*

  Written by Michael Liang - Aug 2nd, 2021

  The following variables are automatically generated and updated when
    changes are made to the Thing

  float t;
  float distance;
  float percentfull;
*/

#include "thingProperties.h"
#include <Arduino_LSM6DS3.h>
#include <Adafruit_SleepyDog.h>

const int trigpin = 11;
const int echopin = 12;
double duration;
double lastdistance;

void setup() {
```

```
22
23    pinMode(trigpin, OUTPUT);
24    pinMode(echopin, INPUT);
25    // Initialize serial and wait for port to open:
26    Serial.begin(9600);
27    // This delay gives the chance to wait for a Serial Monitor without
        blocking if none is found
28    delay(3000);
29
30    while (!IMU.begin()) {
31      Serial.println("Failed to initialize IMU!");
32    }
33    // Defined in thingProperties.h
34    initProperties();
35
36    // Connect to Arduino IoT Cloud
37    ArduinoCloud.begin(ArduinoIoTPreferredConnection);
38    /*
39      The following function allows you to obtain more information
40      related to the state of network and IoT Cloud connection and errors
41      the higher number the more granular information  y o u ll  get.
42      The default is 0 (only errors).
43      Maximum is 4
44    */
45    setDebugMessageLevel(2);
46    ArduinoCloud.printDebugInfo();
47
48  }
49
50  void loop() {
51
52    if (ArduinoCloud.connected() == 0) {
53      Serial.println("Not connected to Arduino Cloud.");
54      ArduinoCloud.begin(ArduinoIoTPreferredConnection);
55    }
56    ArduinoCloud.update();
57    if (IMU.temperatureAvailable()) {
58      // after IMU.readTemperature() returns, t will contain the temperature
        reading
59      IMU.readTemperature(t);
60      digitalWrite(trigpin, HIGH);
61      delayMicroseconds(10);
62      digitalWrite(trigpin, LOW);
63      duration = pulseIn(echopin, HIGH);
64      distance = (duration * (0.03313 + 0.0000606 * t) / 2);
65    }
66    Watchdog.sleep();
67    delay(1000);
68
69  }
```

3 Current Status

The depth sensor has been tested thoroughly and is functioning with high accuracy. The only problem that occurs is that once in a while, the Arduino Nano 33 IoT disconnects from the wifi and needs to be reconnected by resetting the device. Currently, the device is not being used since it is still being improved for usability (wifi disconnections) and quality. Additionally, as a result of the Ubidots API data limitation (used json HTTPS POST REQUESTs), I have relied on the Arduino API to have a constantly running dashboard to monitor the data being received.

4 Future Improvements

As a result of the limited features of the sensor, I'm planning to adding another attachable box to the sensor stand above the pond. This attachable box will feature an LCD display as well as a camera that will have multiple features updated in the future. It will offer object detection to ensure all the fish are in the pond (in case any are eaten or jump out). Once I take more courses in the field of computer vision and object detection in the future, I will begin to work on fish disease detection and implement it into this sensor.