

Universidad Galileo

Gerson Lopez

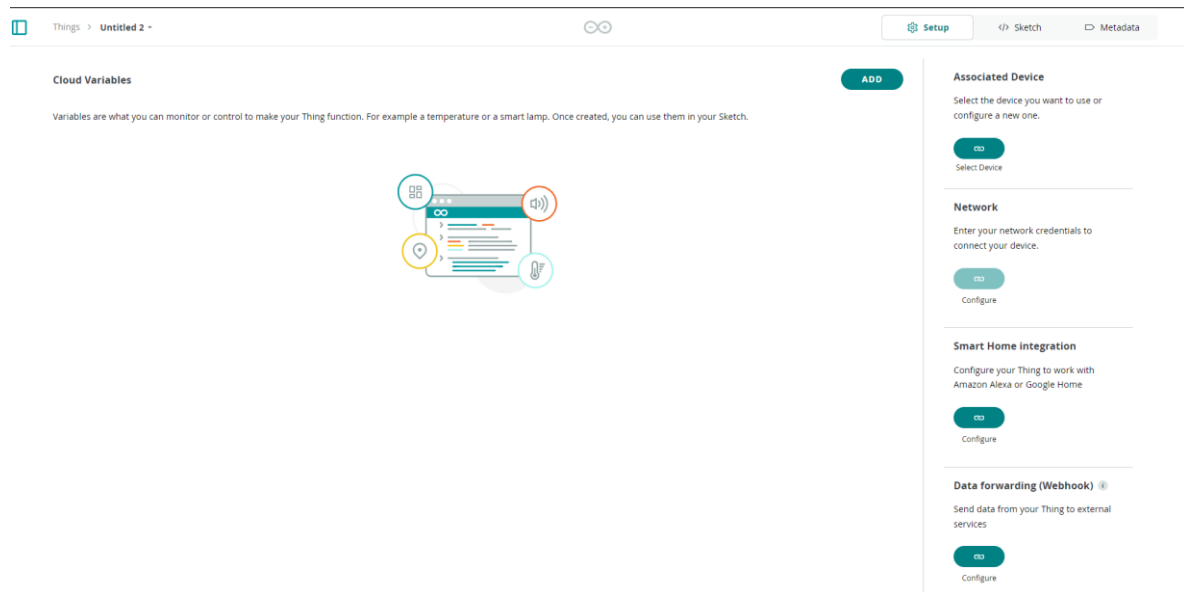
Hardware/Microcontroladores/sistemas embebidos

## **Entrega 2**

**Avances proyecto entorno de programación**

## Arduino cloud para la implementación de sistemas IoT

Arduino Cloud es una plataforma en línea que facilita la creación, gestión y monitoreo de proyectos IoT, permitiendo a los usuarios conectar varios dispositivos compatibles con Arduino, es similar al IDE Arduino sin embargo añade la característica de poder conectarlo con internet para realizar diversas tareas de automatización y control, por eso se eligió esta plataforma para la creación del riego automático ya que Arduino cloud proporciona las herramientas necesarias para implementar un sistema de riego automático eficiente y fácil de gestionar la integración de sensores y el control remoto y la capacidad de monitoreo en tiempo real son una de las características que hacen que este proyecto este a nivel para ser usado por el momento a un nivel del hogar.



Se agrega un Things

Arduino cloud nos permite agregar un “Things” estos representan un dispositivo de control un MCU, para iniciar el proyecto se agrego en el apartado de “**Associated Device**”, se agrego un ESP32 seguido se configuro la red de conexión para el dispositivo, en esta parte también se puede configurar un asistente personal como Alexa, Google home

The screenshot shows the Arduino IoT Cloud dashboard for a project named "Untitled". The "Cloud Variables" section lists five variables:

Name	Last Value	Last Update
<input type="checkbox"/> dht11_humedad float dht11_humedad;	NaN	16 Jun 2024 03:19:27
<input type="checkbox"/> dht11_temperatura float dht11_temperatura;	NaN	16 Jun 2024 03:19:27
<input type="checkbox"/> Estado int estado;	0	16 Jun 2024 03:19:27
<input type="checkbox"/> humedad_p float humedad_p;	100	16 Jun 2024 11:05:10
<input type="checkbox"/> Riego_manual bool riego_manual;	false	16 Jun 2024 03:19:27

The "Associated Device" section shows an ESP32\_IoT device with ID 270a534b-515a-4037-b3f9-0589, Type: uPesy ESP32 Wroom DevKit, and Status: Offline. It includes buttons for "Change" and "Detach".

The "Network" section prompts for network credentials to connect the device, with a "Configure" button.

The "Smart Home Integration" section prompts for configuration with Amazon Alexa or Google Home, with a "Configure" button.

The "Data forwarding (Webhook)" section prompts for sending data to external services, with a "Configure" button.

## Variable a monitorear

Cloud variable, estas son variable ya se de sensores o actuadores que vayamos a utilizar en el proyecto aun que también puede ser “variables virtuales” como lo es la variable **“Riego\_manual”** que es utilizado para poder monitoriar el estado de la configuración de riego que puede estar en automático o bien, en riego manual.

The screenshot shows the Arduino IDE with the following tabs: "Untitled\_jun05a.ino", "ReadMe.adoc", and "thingProperties.h". The code in the "Untitled\_jun05a.ino" tab is as follows:

```

1  #include <DHTesp.h>
2  /*
3   Sketch generated by the Arduino IoT Cloud Thing "Untitled"
4   https://create.arduino.cc/cloud/things/19bebfed-1b04-46da-bde9-a0dfa9d7c6fe
5
6   Arduino IoT Cloud Variables description
7
8   The following variables are automatically generated and updated when changes are made to the Thing
9
10  float dht11_humedad;
11  float dht11_temperatura;
12  float humedad_p;
13  int estado;
14  bool riego_manual;
15
16  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
17  which are called when their values are changed from the Dashboard.
18  These functions are generated with the Thing and added at the end of this sketch.
19  */

```

## Inicio de variable antes declaradas

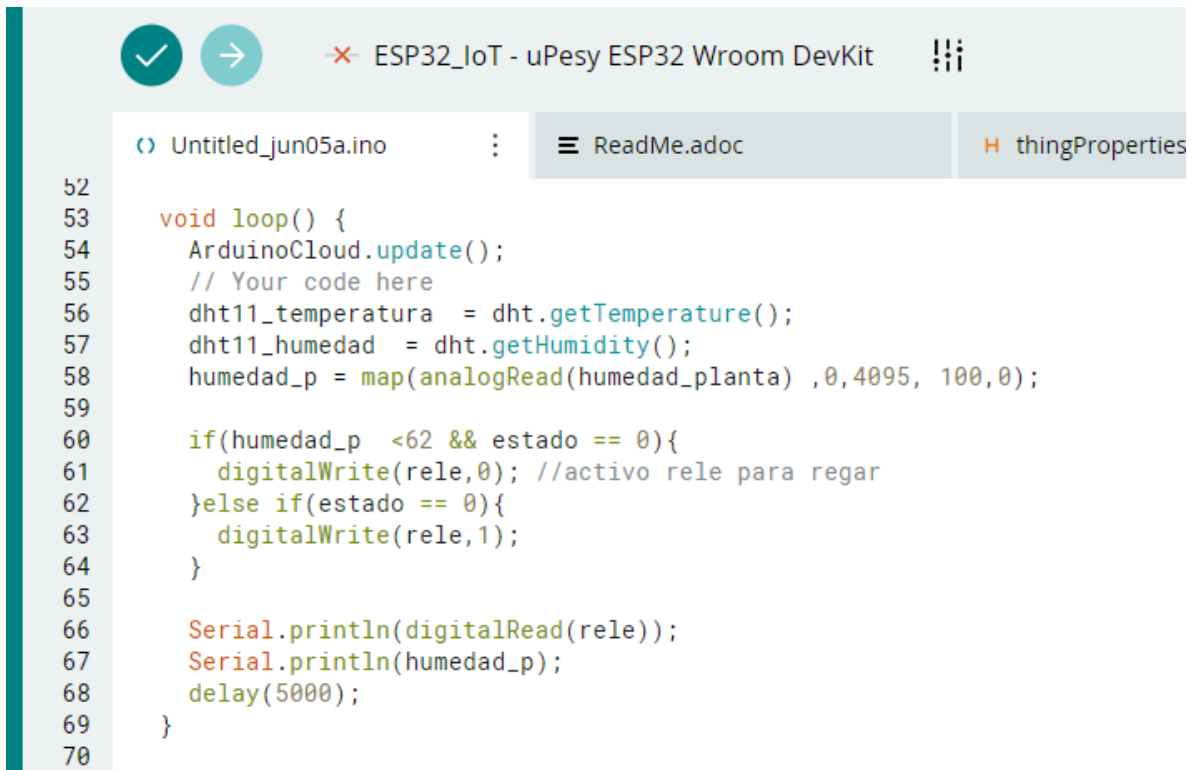
Una vez declaradas las variables en el sketch que nos crea Android cloud ya nos da un esqueleto con todo lo necesario para la comunicación por internet con el ESP32, también a un inicio nos indica que las variables creadas en cloud variables ya están declaradas, es solo de usarlas con lógica para lograr controlar el hardware a nuestra conveniencia.

```
Untitled_jun05a.ino  :  ReadMe.adoc  thingProperties.h  +
25
26 void setup() {
27   // Initialize serial and wait for port to open:
28   Serial.begin(9600);
29   // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
30   delay(1500);
31
32   // Defined in thingProperties.h
33   initProperties();
34
35   // Connect to Arduino IoT Cloud
36   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
37
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 you'll get.
42    The default is 0 (only errors).
43    Maximum is 4
44   */
45   setDebugMessageLevel(2);
46   ArduinoCloud.printDebugInfo();
47   dht.setup(23,DHTesp::DHT11);
48   pinMode(rele, OUTPUT);
49   pinMode(humedad_planta, INPUT);
50
51 }
52
```

## Setup

En el setup inicializamos y configuramos los pines que vamos a utilizar en el proyecto, los pines a utilizar son:

- Pin 22 -> rele
- Pin 36 -> sensor humedad por electrodo
- Pin 23 <- sensor DHT11(humedad-temperatura)



The screenshot shows the Arduino IDE interface. At the top, there's a toolbar with a checkmark, a right arrow, and a red 'X' icon. The title bar reads 'ESP32\_IoT - uPesy ESP32 Wroom DevKit'. Below the title bar, there are tabs for 'Untitled\_jun05a.ino', 'ReadMe.adoc', and 'thingProperties'. The main editor area displays the following C++ code:

```
52
53 void loop() {
54   ArduinoCloud.update();
55   // Your code here
56   dht11_temperatura = dht.getTemperature();
57   dht11_humedad = dht.getHumidity();
58   humedad_p = map(analogRead(humedad_planta) ,0,4095, 100,0);
59
60   if(humedad_p <62 && estado == 0){
61     digitalWrite(rele,0); //activo rele para regar
62   }else if(estado == 0){
63     digitalWrite(rele,1);
64   }
65
66   Serial.println(digitalRead(rele));
67   Serial.println(humedad_p);
68   delay(5000);
69 }
70
```

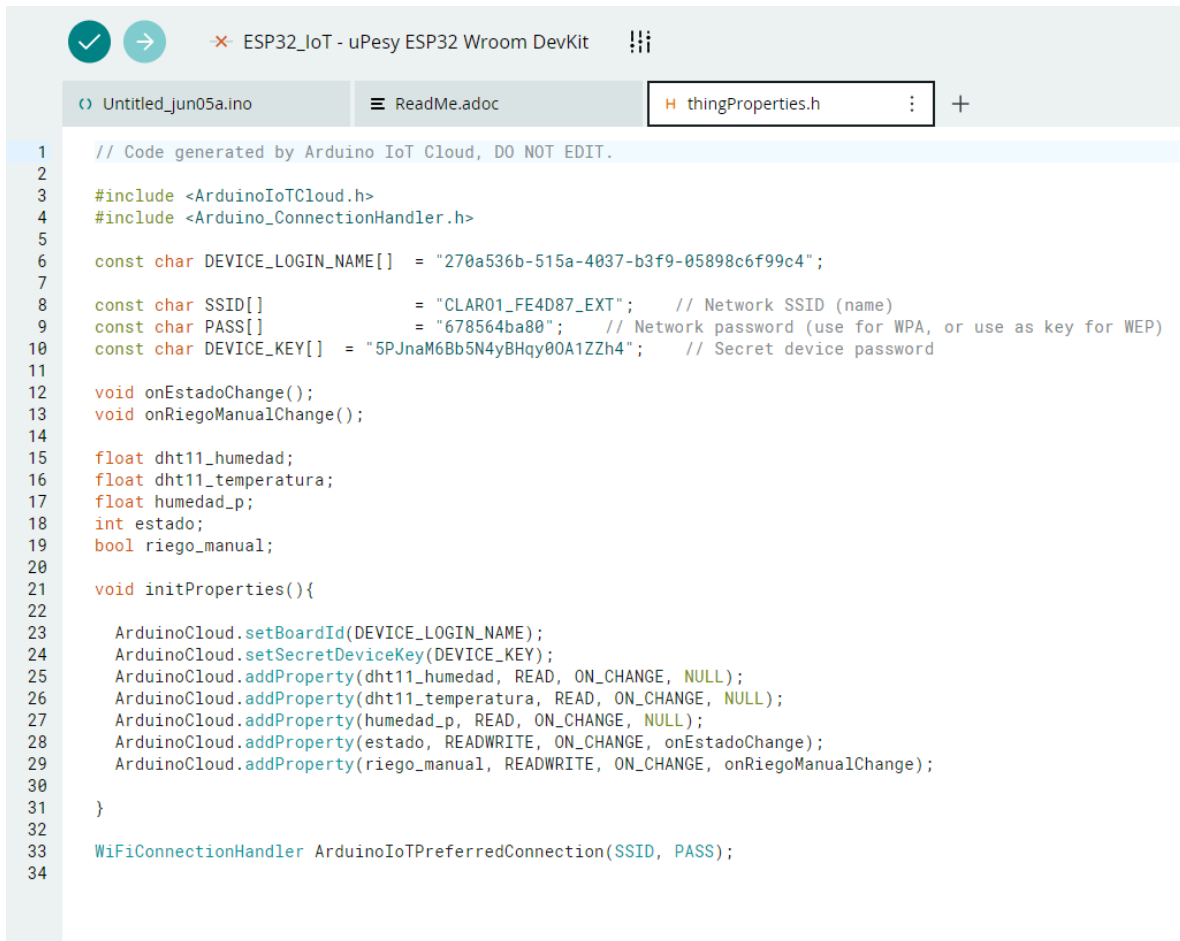
### Loop

En el loop lo que hacemos es actualizar las lecturas de los sensores y en base a estas tomar decisiones para poder encender o apagar el actuador en este caso una bomba de agua, si bien vemos ya Arduino implementa la librería para poder leer de un mejor forma y mas fácil los valores proporcionados por el sensor DHT11 ahora para el sensor insertado en la tierra se usa la función map() para poder mapear el valor leído y asociarlo a un nivel de humedad de la tierra.

```
74
75 void onRiegoManualChange() {
76   // Add your code here to act upon RiegoManual change
77   if(riego_manual){
78     digitalWrite(rele,0);
79     estado = 1;
80   }else{
81     estado = 0;
82   }
83 }
84
85
```

### Subrutina Riego manual

Esta subrutina se encarga de poder tomar el control del riego de forma manual, se colocó para poder tener el control de la bomba en dado caso el sensor deje de funcionar y no haya nadie en casa que pueda apagarlo,



```
1 // Code generated by Arduino IoT Cloud, DO NOT EDIT.
2
3 #include <ArduinoIoTCloud.h>
4 #include <Arduino_ConnectionHandler.h>
5
6 const char DEVICE_LOGIN_NAME[] = "270a536b-515a-4037-b3f9-05898c6f99c4";
7
8 const char SSID[] = "CLAR01_FE4D87_EXT"; // Network SSID (name)
9 const char PASS[] = "678564ba80"; // Network password (use for WPA, or use as key for WEP)
10 const char DEVICE_KEY[] = "5PJnaM6Bb5N4yBHqy00A1ZZh4"; // Secret device password
11
12 void onEstadoChange();
13 void onRiegoManualChange();
14
15 float dht11_humedad;
16 float dht11_temperatura;
17 float humedad_p;
18 int estado;
19 bool riego_manual;
20
21 void initProperties(){
22
23     ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
24     ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
25     ArduinoCloud.addProperty(dht11_humedad, READ, ON_CHANGE, NULL);
26     ArduinoCloud.addProperty(dht11_temperatura, READ, ON_CHANGE, NULL);
27     ArduinoCloud.addProperty(humedad_p, READ, ON_CHANGE, NULL);
28     ArduinoCloud.addProperty(estado, READWRITE, ON_CHANGE, onEstadoChange);
29     ArduinoCloud.addProperty(riego_manual, READWRITE, ON_CHANGE, onRiegoManualChange);
30
31 }
32
33 WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
34
```

## Configuración para conectar a internet el MCU



Dashboards		
Name ↑	Last modified	Sharing with
<input type="checkbox"/> Riego automatizado	12 Jun 2024, 4:35 p.m.	6

Se agrega un dashboard para controlar y monitorizar las variables