ANEXO-C Código Esp32 Controlador

Código Esp32 Controlador y Lora sx1278 Receptor (Código Arduino Cloud)

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/6bcf4f0e-51ec-4ab5-acb8-1f21435a0bb2

  Arduino IoT Cloud Variables description

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

  float humedad2;
  float temperatura2;
  bool botonRiego;
  bool ledDato;
  bool releRiego;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <LoRa.h>
#include <SPI.h>   // Librería SPI para comunicación
```

```
// Pines del LoRa
#define CS 5      // Este pin se puede cambiar por otro GPIO
#define RESET 14   // Este pin se puede cambiar por otro GPIO
#define DIO0 26    // Este pin se puede cambiar por otro GPIO

// Configuración de los LEDs
#define LED_INDICADOR 25  // pin de LED indicador de recepción de datos
#define BOMBA 4         // pin para encender BOMBA de agua cuando hay heladas
#define ELECTROVALVULA 2  // pin para encender ELECTROVALVULA cuando hay heladas


// Configuración de la pantalla LCD
LiquidCrystal_I2C lcd(0x27, 20, 4);  // Dirección I2C y tamaño del display (20x4)


void setup() {
  // Inicia la comunicación serial con una velocidad de 9600 baudios
  Serial.begin(9600);
  // Espera hasta que la comunicación serial esté lista antes de continuar con el programa
  delay(1500);
  while (!Serial);


  // Defined in thingProperties.h
  initProperties();
```

```
 // Connect to Arduino IoT Cloud
 ArduinoCloud.begin(ArduinoIoTPreferredConnection);

 /*
   The following function allows you to obtain more information
   related to the state of network and IoT Cloud connection and errors
   the higher number the more granular information you'll get.
   The default is 0 (only errors).
   Maximum is 4
*/
 setDebugMessageLevel(2);
 ArduinoCloud.printDebugInfo();


 pinMode(12, OUTPUT);            // Configura el pin de la base del transistor 2N2222, el cual enciende el rele de riego LED como salida
 pinMode(LED_INDICADOR, OUTPUT);    // Configura el LED_indicador como salida
 pinMode(BOMBA, OUTPUT);         // Configura el BOMBA como salida
 pinMode(ELECTROVALVULA, OUTPUT);   // Configura el ELECTROVALVULA como salida

 //configuro el estado del led, la bomba y la electrovalvula para cuando inicia el programa
 digitalWrite(LED_INDICADOR, LOW);
 digitalWrite(BOMBA, HIGH);
 digitalWrite(ELECTROVALVULA, HIGH);


 // Iniciar la pantalla LCD
 lcd.init();            // Inicializar el LCD
```

```
 lcd.backlight();        // Encender la luz de fondo
 lcd.setCursor(0, 0);
 lcd.print("Iniciando...");

 // Configurar LoRa sx1278
 LoRa.setPins(CS, RESET, DIO0);

 LoRa.begin(433E6);       // cambia la frecuencia a 433MH

 if (!LoRa.begin(433E6)) {  // evalua si la frecuencia es de 433MH, si no entra al bucle
  lcd.setCursor(0, 1);
  lcd.print("Error LoRa");
  while (1);
 }

/* Configurar parámetros avanzados (opcional, se dejo por defecto)
 LoRa.setSpreadingFactor(7);     // Factor de expansión (6-12, valores más altos =
mayor alcance, pero menor velocidad)
 LoRa.setSignalBandwidth(125E3); // Ancho de banda (7.8kHz a
500kHz)(125,250,500) (mayor ancho =mayor velocidad, pero menor sensibilidad)
 LoRa.setCodingRate4(5);       // Tasa de codificación (5-8) (mayor valor =mayor
robustes, pero menor velocidad)
 LoRa.setTxPower(20);         // Potencia de transmisión (en dBm, 2-20) */


 lcd.setCursor(0, 1);
 lcd.print("LoRa iniciado.");
 delay(2000);
 lcd.clear();
}
```

```
void loop() {
  ArduinoCloud.update();
  // Your code here

  // Verificar si hay datos recibidos
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    String message = "";

  while (LoRa.available()) {
  message = LoRa.readString();
  }

  // Mostrar el mensaje recibido en el monitor serial
  Serial.print("Paquete recibido: ");
  Serial.println(message); //muestra toda la cadena del mensaje recibido en el monitor serie



  // Encender el LED indicador
  digitalWrite(LED_INDICADOR, HIGH);
  delay(200);  // Mantener el LED encendido por 0.2 segundos
  digitalWrite(LED_INDICADOR, LOW);

  // Identificador del mensaje
  if (message.startsWith("Temperatura:")) { // aca escribo el identificador del mensaje para que solo receptor deseado pueda leerlo, ejemplo:Temperatura, 753 o PC35, etc.. (identficador= Temperatura:)
```

```
  // Encontrar el índice donde comienza la temperatura
  int tempStartIndex = message.indexOf("Temperatura:") + 12; // Saltar la plabra "Temperatura:" que tiene 12 caracteres
  int tempEndIndex = message.indexOf("C", tempStartIndex);   // Buscar el índice de "C"

  // Extraer la subcadena de la temperatura y convertirla a float
  String tempStr = message.substring(tempStartIndex, tempEndIndex);
  float temperature = tempStr.toFloat();
  temperatura2 = temperature;

  // Encontrar el índice donde comienza la humedad
  int humStartIndex = message.indexOf("Humedad:") + 9; // Saltar la palabra "Humedad: "
  int humEndIndex = message.indexOf("%", humStartIndex); // Buscar el índice del espacio después de "%"

  // Extraer la subcadena de la humedad y convertirla a float
  String humStr = message.substring(humStartIndex, humEndIndex);
  float humidity = humStr.toFloat();
  humedad2 = humidity;

  //mostrar en monitor serie
   Serial.print("Temperatura extraida: ");
   Serial.print(temperature);
   Serial.println("C  ");
   Serial.print("Humedad extraida: ");
   Serial.print(humidity);
```

```
  Serial.println("% ");


  // Encender el control de helada si la temperatura es menor a 1°C


    if (temperature <= 1) {
    digitalWrite(BOMBA, LOW); //activa la BOMBA (mantiene la encendida la bomba de agua)
    digitalWrite(ELECTROVALVULA, LOW); //activa la ELECTROVALVULA, cerrando el circuito de riego y enviando el agua al circuito para el control de heladas
    lcd.setCursor(0, 3);
    lcd.print("ctrl helada activado");
    }
  // apagar control de helada si la temperatura es mayor a 4°C (se le da 3 grados de histeresis)
  if (temperature >= 4) {
    digitalWrite(BOMBA, HIGH);
    digitalWrite(ELECTROVALVULA, HIGH);
    lcd.setCursor(0, 3);
    lcd.print("ctrl helada apagado ");
    }
  }


  //Mostrar el mensaje en la pantalla LCD
  //lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Recibido:");
  lcd.setCursor(0, 1);
  lcd.print("Temperatura:");
  lcd.print(temperatura2);
  lcd.print(" C ");
```

```
  lcd.setCursor(0, 2);
  lcd.print("Humedad:");
  lcd.print(humedad2);
  lcd.print(" % ");

 }
 }



/*
 Since Botonriego is READ_WRITE variable, onBotonriegoChange() is
 executed every time a new value is received from IoT Cloud.
*/
void onBotonRiegoChange()  {
 // Add your code here to act upon Botonriego change
if (botonRiego == 1) {
  releRiego = 1;         // Cambia el estado del led digital "Riego" a activado (verde)
  digitalWrite(12,1);    // Enciende el pin de la base del transistor 2N2222, el cual enciende el rele de riego
  delay(200);          // Pausa para evitar rebotes
 }

if (botonRiego == 0) {
  releRiego = 0;          // Cambia el estado del led digital "Riego" a apagado (rojo)
  digitalWrite(12, 0);    // apaga el pin de la base del transistor 2N2222, el cual apaga el rele de riego
  delay(200);
 }
}
```