

ANEXOS

CÓDIGO FUENTE

```

#include "arduino_secrets.h"
/*
  Sketch generated by the Arduino IoT Cloud Thing "Dispensador"
  https://create.arduino.cc/cloud/things/1435b8d9-dce8-49e7-b179-cf83a38f181b

  Arduino IoT Cloud Variables description

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

  float tempDHT;
  int counter;
  CloudSchedule schedule_test;
  bool boton1;
  CloudTime time_read;

  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.
*/

//Vincular archivo creado por Arduino Cloud
#include "thingProperties.h"

//Librerias para servo y RTC
#include <Servo.h>
#include <RTCZero.h>

//Temperature sensor configuration
//Configuracion de sensor de temperatura
#include <DHT.h>
#define DHTPIN 11
#define DHTTYPE DHT11

//OLED Display configuration
//Configuracion de sensor de display
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

DHT dht(DHTPIN, DHTTYPE);
RTCZero alarm;
Servo myservo;
CloudTime time_read;

int pirSensor = 7;
const int buzzer = 5;
int pos = 0;
int inputPin_2 = 2;
int inputPin_3 = 3;
int inputPin_4 = 4;
int button2 = 0;
int button3 = 0;
int button4 = 0;
int segundo_alarma = 0;
bool toggle;

void setup() {

```

```

if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // No proceder si no se inicializa el display
}

Serial.begin(9600);

while(!Serial); //Espera por un puerto serial
delay(1500);

myservo.attach(6);
pinMode(pirSensor, INPUT);
pinMode(buzzer, OUTPUT);

// Configuraci3n de pins para pulsadores
pinMode(inputPin_2, INPUT_PULLDOWN);
pinMode(inputPin_3, INPUT_PULLDOWN);
pinMode(inputPin_4, INPUT_PULLDOWN);

display.clearDisplay(); //Limpiar el display al iniciar
display.setRotation(2); //Rotar el display 180 grados

dht.begin();

// Definidos en thingProperties.h
initProperties();

// Conectar a 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 youâ€™ll get.
    The default is 0 (only errors).
    Maximum is 4
*/
//setDebugMessageLevel(0);
//ArduinoCloud.printDebugInfo();
alarm.begin();
}

//Funcion para mostrar texto principal
void setMainText(){
    display.clearDisplay();
    display.setTextSize(3);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(20, 20);
    display.cp437(true);
}

//Funcion para mostrar mensaje de error
void mostrarError(){
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(20,20);
    display.print("Espere...");
    tone(buzzer,500);
    display.display();
    delay(2000);
    noTone(buzzer);
}

//Funcipn de configuraci3n del reloj
void reloj(){
    int Hours    = alarm.getHours()-3; //-3 por area horaria de Chile
    int Minutes = alarm.getMinutes();

```

```

int Seconds = alarm.getSeconds();
setMainText();

//Evita que se muestren numeros negativos en la hora
if (Hours < 0) {
    Hours += 24;
}

//Incluir un 0 antes de las horas del 0 al 9
if (Hours < 10) {
    display.print("0");
    display.print(Hours);
} else display.print(Hours);

//Incluir un 0 antes de los minutos del 0 al 9
if (Minutes < 10) {
    display.setCursor(72,20);
    display.print("0");
} else
    display.setCursor(72,20);
    display.print(alarm.getMinutes());
    display.setCursor(54,20);
    display.print(":");
    display.display();
}

//Funcion para mostrar temperatura
void showTemp(){
display.clearDisplay();
setMainText();
display.print(tempDHT);
display.setCursor(110,20);

display.setTextSize(2);
display.write("C");
display.display();
delay(2000);
}

//Funcion para activar el mecanismo de dispensacion
void dispensar(){
    setMainText();

    tone(buzzer,4000); delay(100);
    tone(buzzer,5000); delay(100);
    tone(buzzer,5500); delay(100);
    noTone(buzzer);
    delay(500);

//El motor rota en tramos de 45 grados
// ya que el maximo del servo SG90 son 180 grados
Serial.println(pos);
if (pos >= 180) {
    pos = 0;
} else {
    pos += 45;
}

myservo.write(pos);

//Mensajes de debug
Serial.print("Servo position: ");
Serial.println(pos);

display.display();

```

```

    delay(10);
}

void activarAlarma(){
    setMainText();
    display.write("ALARMA");
    display.display();
    //sonido de alarma
    tone(buzzer,3000);    delay(100);
    noTone(buzzer);       delay(100);
    tone(buzzer,2000);    delay(100);
    noTone(buzzer);       delay(100);
    tone(buzzer,1000);    delay(100);
    noTone(buzzer);       delay(1000);
    Serial.println("alarma activada");
}

void loop() {
    ArduinoCloud.update();

    tempDHT = dht.readTemperature();

    button2 = digitalRead(inputPin_2);
    button3 = digitalRead(inputPin_3);
    button4 = digitalRead(inputPin_4);

    /*Si se activa el scheduler en Arduino Cloud se llama
    la funcion de alarma*/
    if (schedule_test.isActive()){
        activarAlarma();
        if (button2 == HIGH) {
            dispensar();
            display.clearDisplay();
            display.write("OK");
            display.display();
            //delay para no presionar 2 veces el botÃ³n
            delay(90000);}
    }

    if(ArduinoCloud.connected()){
        time_read = ArduinoCloud.getLocalTime();
    }
    reloj();
    // Opciones
    if (button2 == HIGH && !schedule_test.isActive()) mostrarError(); //
    if (button3 == HIGH) dispensar(); //Para probar el mecanismo
    if (button4 == HIGH) showTemp(); //Ver la temperatura
}

////////////////////////////////////
////////////////////////////////////

/*
    Since Boton2 is READ_WRITE variable, onBoton2Change() is
    executed every time a new value is received from IoT Cloud.
*/
void onBoton1Change() {
    // Add your code here to act upon Boton2 change
    dispensar();
}
/*
    Since TempDHT is READ_WRITE variable, onTempDHTChange() is
    executed every time a new value is received from IoT Cloud.
*/

```

```
*/  
void onTempDHTChange() {  
    ArduinoCloud.update();  
    // Add your code here to act upon TempDHT change  
}  
  
void onScheduleTestChange() {  
    // Add your code here to act upon ScheduleTest change  
}
```

Responses Overview Active

Responses

64



Average Time

01:35



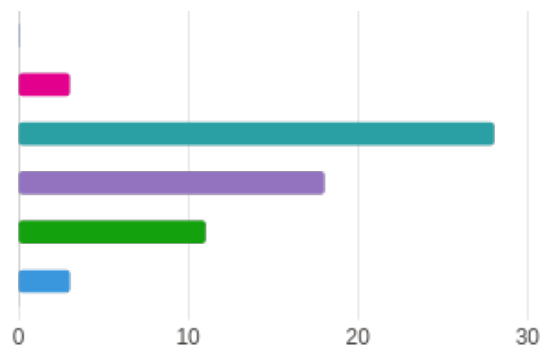
Duration

24 Days



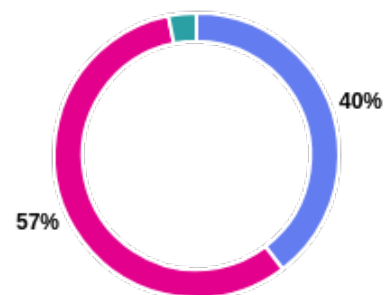
1. Por favor indique el rango de edad del usuario final del producto

0-10 años	0
11-20 años	3
entre 21 y 40 años	28
entre 41 y 60 años	18
entre 61 y 80 años	11
sobre 80 años	3



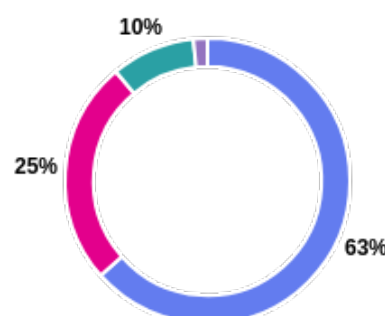
2. Por favor indique el sexo del usuario final del producto

Masculino	25
Femenino	36
Prefiero no responder / Otro	2



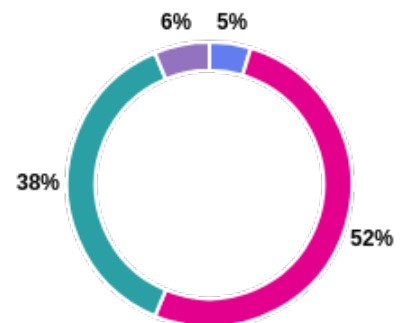
3. ¿Qué cantidad de medicamentos/suplementos diferentes necesita el consumir diariamente el usuario final?

de 1 a 3	40
de 4 a 6	16
de 7 a 10	6
más de 10	1



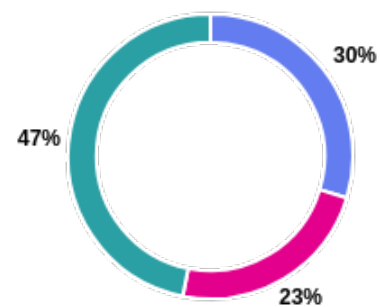
4. ¿Olvida tomar sus medicamentos?

● Siempre o el usuario final no está capacitado para medicarse sólo	3
● A menudo	33
● En raras ocasiones	24
● Nunca	4



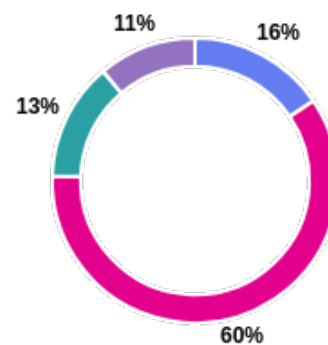
5. ¿Utiliza algún organizador de medicamentos actualmente?

● Si	19
● Lo he considerado	15
● No	30

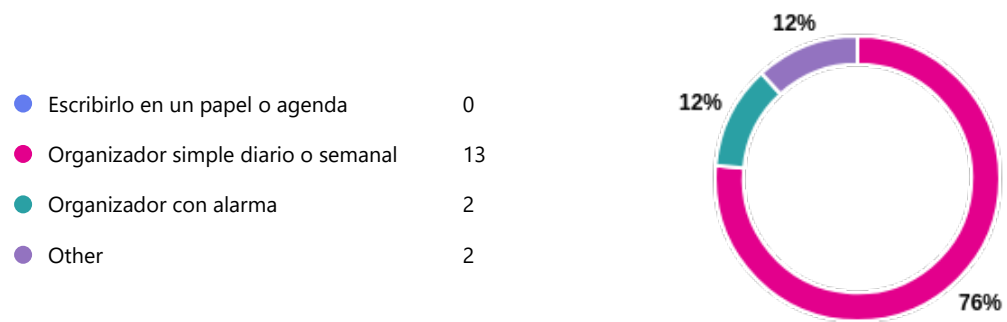


6. ¿Está usted interesado en la adquisición de un dispensador de medicamentos?

● Muy Interesado	7
● Interesado	27
● Poco Interesado	6
● No Interesado	5



7. ¿Qué tipo de organizador de medicamentos ha utilizado?



8. ¿Cuánto estaría dispuesto a pagar por la adquisición de un pastillero inteligente?

