UNIVERSIDAD TECNOLÓGICA DE TIJUANA

TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA

MATTER:

APLICACIONES DE IOT

PROFESSOR:

DR. CESAR ORTEGA CORRAL

STUDENT(s):

ARISTA PEREZ GRACIELA

DIAZ ESCALANTE JOSÉ ÁNGEL

SAMONTE MERCADO JEREMY

GROUP:

5A BIS

TOPIC:

UPDATED IoT Project Proposal and sensor tests

ACTIVITY:

Assignment #2

DATE:

THURSDAY, JUNE 13, 2024

Title: Smart Inventory

Area: Logistics and Warehouse Management

Documentation Structure

- Objective

The objective of this project is to design and implement a Smart Inventory Management system using IoT sensors, mobile app, and website to improve the efficiency and accuracy of inventory management. Integrating sensors for the project will provide a comprehensive, real-time data useful for inventory monitoring and management. Our goal is to optimize inventory management enabling us to track stock levels, facilitating timely restocking decisions and overall processes

- Introduction

Managing inventory is a critical task for businesses, but it can also be time-consuming and prone to errors. By making use of IoT technology, we can create a system that automates many aspects of inventory management, improving accuracy and freeing up staff to focus on other tasks.

- Problem statement

Traditional inventory management methods rely on manual counting and tracking, which can lead to errors and inefficiencies. It can also be difficult to track inventory levels in real-time and respond quickly to changes in demand.

- Solution proposal

To address these challenges, we propose a Smart Inventory Management system that uses IoT sensors and a mobile app to provide real-time visibility into inventory levels and alerts when inventory levels reach a certain threshold. The system will also include a website that stores historical data and provides user management capabilities.

- Table of Roles:

MEMBER	ROLE, ACTIVITY AND RESPONSIBILITIES
Arista Pérez Graciela	Front-end developer in the activity of mobile development. IoT project participant.
Díaz Escalante José Ángel	Back-end developer in the activity of web applications. IoT project participant.
Samonte Mercado Jeremy	Front-end developer in the activity of mobile development. IoT project participant.

DEVICES: Describe the physical layer and the devices involved.

The following devices will be used in this system:

- **ESP32**: A low-cost microcontroller with built-in WiFi and Bluetooth capabilities. It will serve as the hub for the IoT network and communicate with the weight sensors.

- **Weight Sensor**: A sensor that measures the weight of inventory items. It will communicate with the ESP32 to provide real-time data on inventory levels.
- **Modulo HX711:** This working as a converter for the weight sensor.
- **LED / OLED Display:** Displays the current weight on top of the weight sensor.
- **Humidity Sensor:** To maintain control of the quality of the product.
- **RFID Tags and Readers**: RFID tags and readers will be attached to inventory items and used to track their movement throughout the warehouse. This can help prevent theft and improve security, as well as providing valuable data on how the warehouse is being used.

CONNECTIVITY: Explain the protocol (or protocols) you will use.

ABSTRACTION: What will be shown in the user interface (dashboards)

The user interface will show real-time data on inventory levels, with a dashboard that displays the weight of each inventory item. Users can set custom thresholds for alerts, and the mobile app will notify them when inventory levels fall below those thresholds. The mobile app will also provide data visualization, allowing users to view inventory levels over time in graph or chart format.

APPLICATION: Updated description and general diagram of the end-to-end system.

The end-to-end system will consist of three main components:

- **Physical Layer**: The physical layer consists of the IoT devices, and sensors. The ESP32 microcontroller will serve as the hub for the IoT network, communicating with the weight sensor readers to gather data on inventory levels and movements.
- **Mobile App**: The mobile app will receive data from the ESP32 microcontroller and display real-time data on inventory levels. Users can view the weight of each inventory item, receive alerts when inventory levels fall below custom barriers, and view data visualizations over time.
- **Website**: The website will store historical data on inventory levels, allowing users to view trends and patterns over time. The website will also provide user management capabilities, allowing control access to inventory data for different users.

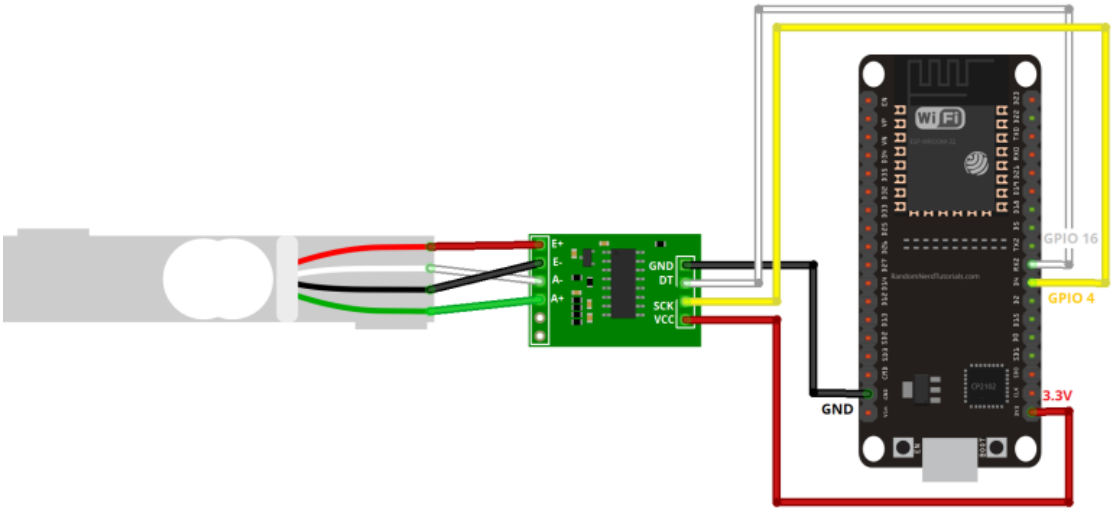
DEVICE Layer

1. **Weight Sensor Load Cell 20 kg with HX711**

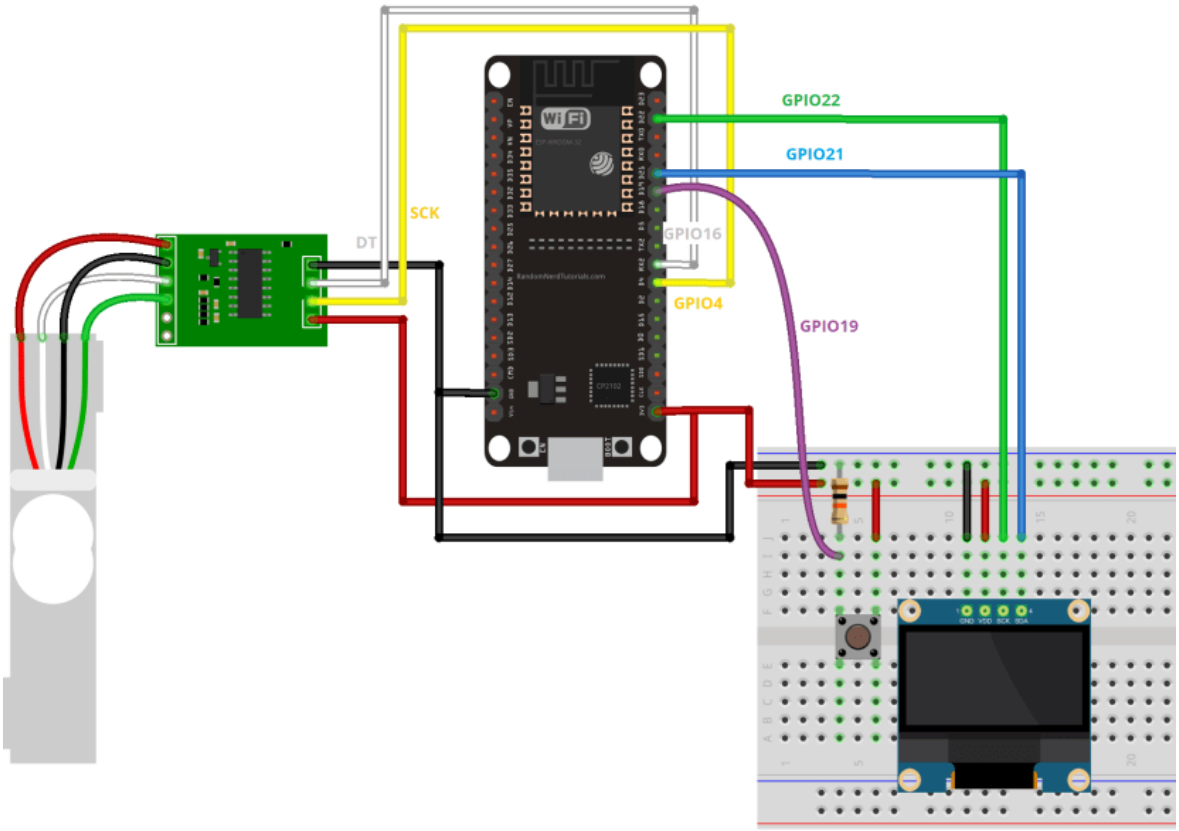
   The 5kg load cell with the HX711 amplifier is a commonly used set for measuring weight in Arduino and ESP32 applications. Measurement of structural element deformation through strain gauges. Typically made of aluminum or stainless steel.

2. **Circuit**

   - **Calibrating (ESP32 with Load Cell)**



- Weighing



- Code 1

```
/*Código para calibrar el sensor*/
// Calibrando la célula de carga
#include <Arduino.h>
#include "soc/rtc.h"
#include "HX711.h"

// Conexión del circuito HX711
const int LOADCELL_DOUT_PIN = 18;
const int LOADCELL_SCK_PIN = 4;

HX711 scale;
```

```
void setup() {
  Serial.begin(115200);
  rtc_cpu_freq_config_t config;
  rtc_clk_cpu_freq_get_config(&config);
  rtc_clk_cpu_freq_to_config(RTC_CPU_FREQ_80M, &config);
  rtc_clk_cpu_freq_set_config_fast(&config);
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
}

void loop() {
  if (scale.is_ready()) {
    scale.set_scale();
    Serial.println("Retire cualquier peso de la balanza.");
    delay(5000);
    scale.tare();
    Serial.println("Tarea completa...");
    Serial.println("Coloque un peso conocido en la balanza...");
    delay(5000);
    long reading = scale.get_units(10);
    Serial.print("Resultado: ");
    Serial.println(reading);
  }
  else {
    Serial.println("HX711 no encontrado.");
  }
  delay(1000);
}

//El factor de calibración será la (lectura)/(peso conocido)
```

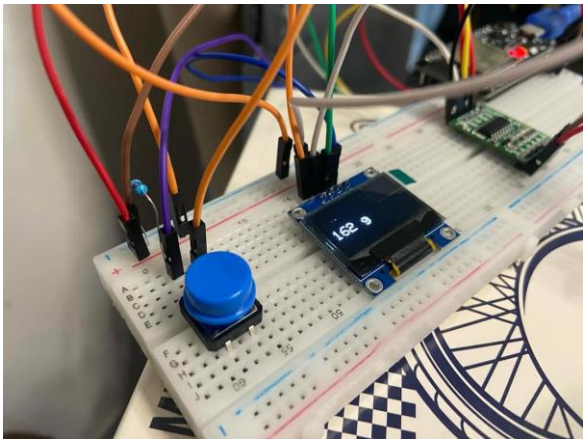
- Code 2

```
#include <Arduino.h>
#include "HX711.h"
#include "soc/rtc.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Pushbutton.h>

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 18;
const int LOADCELL_SCK_PIN = 4;
```

```
HX711 scale;
int reading;
int lastReading;
//REPLACE WITH YOUR CALIBRATION FACTOR
#define CALIBRATION_FACTOR -71.739

//OLED Display
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//Button
#define BUTTON_PIN 19
Pushbutton button(BUTTON_PIN);

void displayWeight(int weight){
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Peso:");
  display.display();
  display.setCursor(0, 30);
  display.setTextSize(2);
  display.print(weight);
  display.print(" ");
  display.print("g");
  display.display();
}

void setup() {
  Serial.begin(115200);
  rtc_cpu_freq_config_t config;
  rtc_clk_cpu_freq_get_config(&config);
  rtc_clk_cpu_freq_to_config(RTC_CPU_FREQ_80M, &config);
  rtc_clk_cpu_freq_set_config_fast(&config);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Error en la inicialización del SSD1306"));
    for(;;);
```

```
  }
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);

  Serial.println("Inicializando la balanza");
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);

  scale.set_scale(CALIBRATION_FACTOR);   // este valor se obtiene calibrando la
balanza con pesos conocidos; vea el README para más detalles
  scale.tare();               // reiniciar la balanza a 0
}

void loop() {
  if (button.getSingleDebouncedPress()){
    Serial.print("Tara...");
    scale.tare();
  }

  if (scale.wait_ready_timeout(200)) {
    reading = round(scale.get_units());
    Serial.print("Peso: ");
    Serial.println(reading);
    if (reading != lastReading){
      displayWeight(reading);
    }
    lastReading = reading;
  }
  else {
    Serial.println("HX711 no encontrado.");
  }
}
```

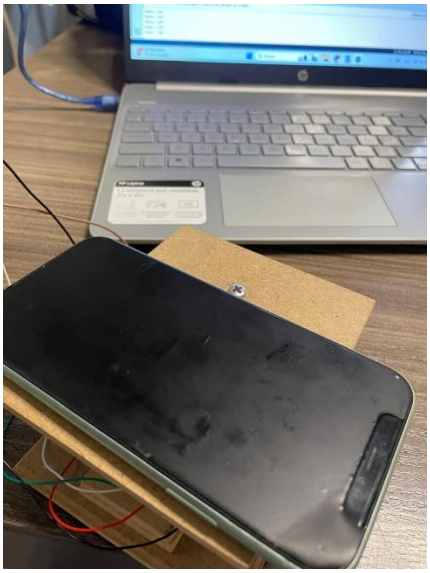
- Measure different levels within the sensor’s range by conducting several experiments.

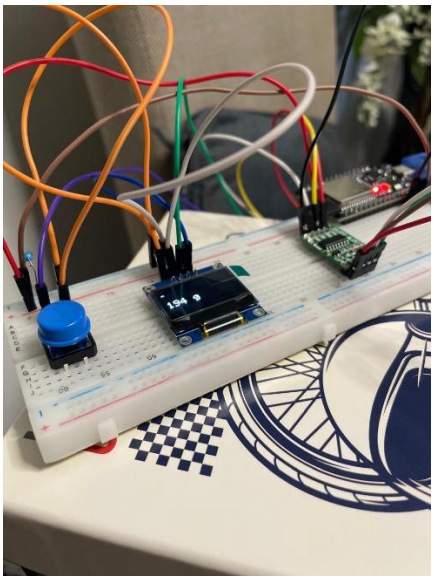


```
Peso: 164
Peso: 163
Peso: 162
Peso: 162
Peso: 162
Peso: 162
Peso: 162
```

Tabla resumen

Características	iPhone 12
Sistema operativo	IOS 14
Colores	Blanco, negro, azul, verde y rojo
Dimensiones	146.7 x 71.5 x 7.4 mm
Peso	**164 gramos**





```
Peso: 194
Peso: 193
Peso: 193
Peso: 194
Peso: 194
Peso: 194
Peso: 194
```

194 gramos

El iPhone 11 es el modelo estándar de esta versión y cuenta con una pantalla de 6.06 pulgadas. Tiene unas dimensiones de 150,9 x 75,7 x 8,3 milímetros y un peso de **194 gramos**. 21 dic 2022

PC Componentes
https://www.pccomponentes.com › tamao-y-medidas-ip...

Tamaño y medidas iPhone 11 (normal, pro y pro max)