RFID BASED INVENTORY MANAGEMENT SYSTEM.

What are RFID'S:

RFID (Radio Frequency Identification) is a wireless technology used for identifying and tracking objects or people using radio waves. It works by storing data on a small electronic device known as an **RFID tag**, which can be attached to or embedded in various items. The RFID tag contains a **microchip** for storing information and an **antenna** for transmitting the data to an RFID **reader**.

There are two main types of RFID tags:

1. **Passive RFID tags**: These don't have an internal power source (battery). They rely on the radio energy transmitted by the RFID reader to power the tag and send back data. They are cheaper but have shorter read ranges (up to a few meters).
2. **Active RFID tags**: These contain their own power source, typically a battery, allowing them to transmit signals over longer distances (up to 100 meters or more). They are more expensive than passive tags.

RFID Components

- **RFID Tag**: The device that stores data.
- **RFID Reader**: A device that emits radio waves to read the tag's data.
- **RFID Antenna**: A part of both the reader and the tag, used to transmit and receive radio signals.

Common Uses of RFID

- **Inventory management**: Tracking products in warehouses and stores.
- **Access control**: Managing entry to secure areas with RFID cards or badges.
- **Library systems**: Managing books, as seen in RFID-based library management systems.
- **Transportation**: In contactless payments and toll systems.

RFID systems are widely used due to their ability to scan multiple items simultaneously, without the need for line-of-sight like barcode systems.

1) **Which RFID tag is beneficial for our use ?**

Here are the four category i am considering in my mind for the our management system
Cost Efficiency: Passive RFID tags are cheaper, making them more suitable for tracking large numbers of items typically found in inventory systems.

1. **Adequate Range**: The read range of passive tags (a few meters) is usually sufficient for warehouse or store environments where items are scanned relatively close to the reader.
2. **Long Lifespan**: Since passive RFID tags don't have batteries, they last longer and don't require maintenance, unlike active tags, which have a finite battery life.
3. **Bulk Scanning**: Passive RFID tags can still be read in bulk by readers, allowing efficient scanning of multiple items at once.

Active RFID tags are typically used in systems where you need long-range tracking (like tracking containers over long distances) or for items that require real-time location tracking, which isn't usually necessary in standard inventory management.

RFID BASED INVENTORY MANAGEMENT SYSTEM.

In summary, **passive RFID tags** are the most practical and cost-effective solution for an inventory management system.

In conclusion : by considering the major factors such as the Adequate Range , long lifespan , bulk scanning, cost We come to conclude that the active type of RFIDs are the best for the use of our project of inventory management system.

Which Active RFID should we use as there are many of them available ranging from size to cost from plastic type to metallic to non-metallic to high temperature tags ?
-The one who is cost efficient but effective and has small size must be or will be our top priority .

Passive RFID tags available in market :
1) Mifare RC522 RFID Sensor
Cost ;~100
Operating current: 13~26mA/10~13mA (current)
Operating voltage : 2.5V~3.3V
Datasheet: http://www.handsontec.com/dataspecs/RC522.pdf

2) PN532 NFC RFID
Cost:~280 INR
Operating voltage 2.7 to 5.5 V
Operating current: 25mA (current)
Datasheet: https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf

3) RDM6300
Cost:~290 INR
Operating voltage 5Vdc
Operating current: <50mA
Datasheet: https://www.handsontec.com/dataspecs/module/RDM6300.pdf

4) Mfre630 RFID
cost :~390 INR
Operating voltage: DC 3.3V
Operation current : 13-26mA
Datasheet: https://www.nxp.com/docs/en/data-sheet/MFRC630.pdf

5) PN533
cost :~180 INR
Operating voltage: 3.6 V
Operation current : 30mA
Datasheet: https://www.nxp.com/docs/en/nxp/data-sheets/PN5331B3HN.pdf

6) **SL030**:
cost :~19 USD
Operating voltage: 3.6 V
Operation current : 5mA
Datasheet: https://cdn.soselectronic.com/productdata/43/02/b5c1f728/sl030.pdf

RFID BASED INVENTORY MANAGEMENT SYSTEM.

7) SM130 RFID Module
cost :~180
Operating voltage: 5 V
Operation current : 180mA
Datasheet: https://www.sparkfun.com/datasheets/Sensors/ID/SM130.pdf

And many more …..

But which one will be the most effect for us ??

So here is the answer:

PN5180 would be the best option. Here's why:

- **Longer read range**: This is useful for scanning items from a distance without needing to be very close to the reader.
- **Multi-protocol support**: It can handle various RFID tags, making it versatile for different inventory types.
- **Better performance**: It's a more advanced module than the RC522 and PN532, offering faster and more reliable tag reading.

In an RFID-based inventory management system, we have decided on the most important component to use. However, to interface it with a microcontroller, which one will be the best and why?

1. ESP32:

- **Why**:
  - Built-in **Wi-Fi** and **Bluetooth**, making it ideal for sending data directly to a server.
  - Plenty of GPIO pins for RFID reader (PN5180, MFRC630) interfacing.
  - Supports SPI and I2C, commonly used with RFID modules.
  - Ideal for low-power, continuous operations.

2. Raspberry Pi Pico W:

- **Why**:
  - Has **Wi-Fi** capability for server connectivity.
  - Fast, with programming options like Python or C++.
  - Great for more complex processing needs with RFID.
  - Supports SPI/I2C for RFID communication.

3. NodeMCU (ESP8266):

- **Why**:
  - Cheaper than ESP32 but with **Wi-Fi** for data transfer.
  - Suitable for simpler projects or budget constraints.
  - Supports SPI/I2C for RFID interfacing.

4. Arduino MKR1000:

- **Why**:
    - **Built-in Wi-Fi** for server communication.
    - Compatible with Arduino's vast library of RFID modules.
    - SPI and I2C support for RFID.
    - Ideal for simpler, Arduino-based applications needing network connectivity.

5. Particle Photon:

- **Why**:
    - Comes with built-in **Wi-Fi** and cloud integration, perfect for IoT projects.
    - Easy to set up for sending data to a server.
    - SPI support for interfacing with RFID modules like MFRC630.
    - Offers good cloud-based management for real-time data.

6. STM32 (with Wi-Fi module):

- **Why**:
    - Powerful processing capability for handling large datasets.
    - Can interface with an external **Wi-Fi module** (like ESP8266 or ESP32 for Wi-Fi).
    - SPI/I2C support for RFID modules.
    - Suitable for industrial-grade inventory systems requiring high performance.

7. Arduino Uno (with Wi-Fi Shield):

- **Why**:
    - If you want to stay within the familiar **Arduino ecosystem**, adding a **Wi-Fi shield** allows for server communication.
    - Plenty of libraries available for RFID modules.
    - Good for smaller-scale projects but might be limited in memory and performance for more advanced systems.

8. Teensy 4.1:

- **Why**:
    - High processing power with **Ethernet** support (Wi-Fi with an external module).
    - Works well for RFID interfaces via SPI/I2C.
    - Suitable if you require high-speed performance and more complex control for your inventory management system.

9. Adafruit Feather M0 Wi-Fi:

- **Why**:
    - **Built-in Wi-Fi** for sending data to a server.
    - Low-power and lightweight, making it ideal for embedded systems.
    - SPI and I2C interfaces for RFID reader integration.
    - Great for smaller, power-efficient inventory management projects.

Conclusion :

RFID BASED INVENTORY MANAGEMENT SYSTEM.

ESP32 remains the top recommendation due to its combination of **built-in Wi-Fi**, **processing power**, **GPIO flexibility**, and **cost-effectiveness** for interfacing with RFID readers and sending data to a server

Programing platform:

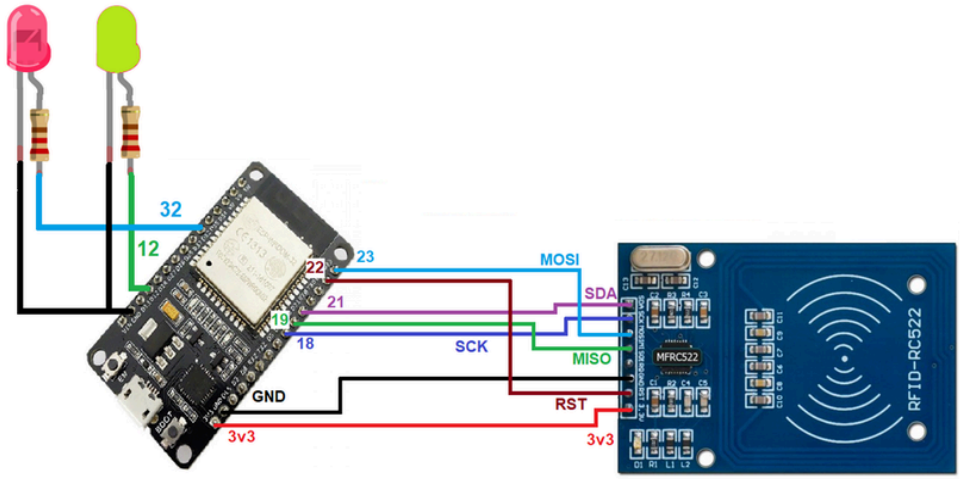
Arduino IDE: Simple and beginner-friendly, with rich libraries for RFID modules and Wi-Fi integration ,
And we also have the experience on it as it was a part of our curricular activities

,

IMPLEMENTATION :

Now we have to interface the our RFID module with our decided MUC and have to right the required code for that

Here is the implementation diagram ;



Now after the connection part of the ESP 32 with our RFID we have to program for it in our arduino ide
:

```
#include "PN5180_Firmware.h"

#define PN5180_RST_PIN      17
#define PN5180_BUSY_PIN     16
#define PN5180_REQ_PIN      4
#define PN5180_NSS_PIN      5
#define STM32_MOSI_PIN      23
#define STM32_MISO_PIN      19
#define STM32_SCK_PIN       18

PN5180_Firmware Pn5180(PN5180_RST_PIN, PN5180_BUSY_PIN, PN5180_REQ_PIN, PN5180_NSS_PIN,
STM32_MOSI_PIN, STM32_MISO_PIN, STM32_SCK_PIN);

void setup() {
  Serial.begin(115200L);
```

```
  Pn5180.Begin();
}

void loop() {
  Serial.println("Press Any Key To Dump PN5180 Info!");

  Serial.flush();

  while (!Serial.available());

  while (Serial.available()) {
    Serial.read();
  }

  Serial.flush();

  Pn5180.End();
  Serial.end();

  Serial.begin(115200L);
  Pn5180.Begin();

  uint8_t MajorVersion, MinorVersion;
  uint8_t* DieId = (uint8_t*)malloc(PN5180_DL_GETDIEID_DIEID_LEN * sizeof(uint8_t));

  Pn5180.GetFirmwareVersion(&MajorVersion, &MinorVersion);
  Serial.print("[Info] Firmware Version "); Serial.print(MajorVersion, HEX);  Serial.print(".");
Serial.println(MinorVersion, HEX);

  Pn5180.GetDieID(DieId);
  Serial.print("[Info] DieID : "); Pn5180.PrintHex8(DieId, PN5180_DL_GETDIEID_DIEID_LEN);
  free(DieId);

  PN5180_DOWNLOAD_INTEGRITY_INFO_T IntegrityInfo;
  Pn5180.CheckIntegrity(&IntegrityInfo);
  Serial.print("[Info] Integrity->FunctionCodeOk : "); Serial.println(IntegrityInfo.FunctionCodeOk);
  Serial.print("[Info] Integrity->PatchCodeOk : "); Serial.println(IntegrityInfo.PatchCodeOk);
  Serial.print("[Info] Integrity->PatchTableOk : "); Serial.println(IntegrityInfo.PatchTableOk);
  Serial.print("[Info] Integrity->UserDataOk : "); Serial.println(IntegrityInfo.UserDataOk);

  PN5180_DOWNLOAD_SESSION_STATE_INFO SessionStateInfo;
  Pn5180.CheckSessionState(&SessionStateInfo);
  Serial.print("[Info] SessionState->LifeCycle : "); Serial.println(SessionStateInfo.LifeCycle);
  Serial.print("[Info] SessionState->SessionState : "); Serial.println(SessionStateInfo.SessionState);
}
```

RFID BASED INVENTORY MANAGEMENT SYSTEM.

This is the code for the getting the values of the RFID not we have to create some functions that will create / delete the input as per our need

```
void addEntity() {
  inventoryCount++;
  Serial.print("[Info] Entity added. Total inventory count: ");
  Serial.println(inventoryCount);
}

void deleteEntity() {
  if (inventoryCount > 0) {
    inventoryCount--;
    Serial.print("[Info] Entity deleted. Total inventory count: ");
    Serial.println(inventoryCount);
  } else {
    Serial.println("[Warning] Inventory is already empty, no entity to delete.");
  }
}
```

Now we get the addition and subtraction function but now we have to upload this code to the server (for example firebase)

Example code to me merged with original one

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* firebaseHost = "your_project_id.firebaseio.com";
const char* firebaseAuth = "your_firebase_database_secret";

void setup() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
}

void pushDataToFirebase(const char* entityID, int count) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = String("https://") + firebaseHost + "/inventory/" + entityID + ".json?auth=" + firebaseAuth;

    http.begin(url);
    http.addHeader("Content-Type", "application/json");
```

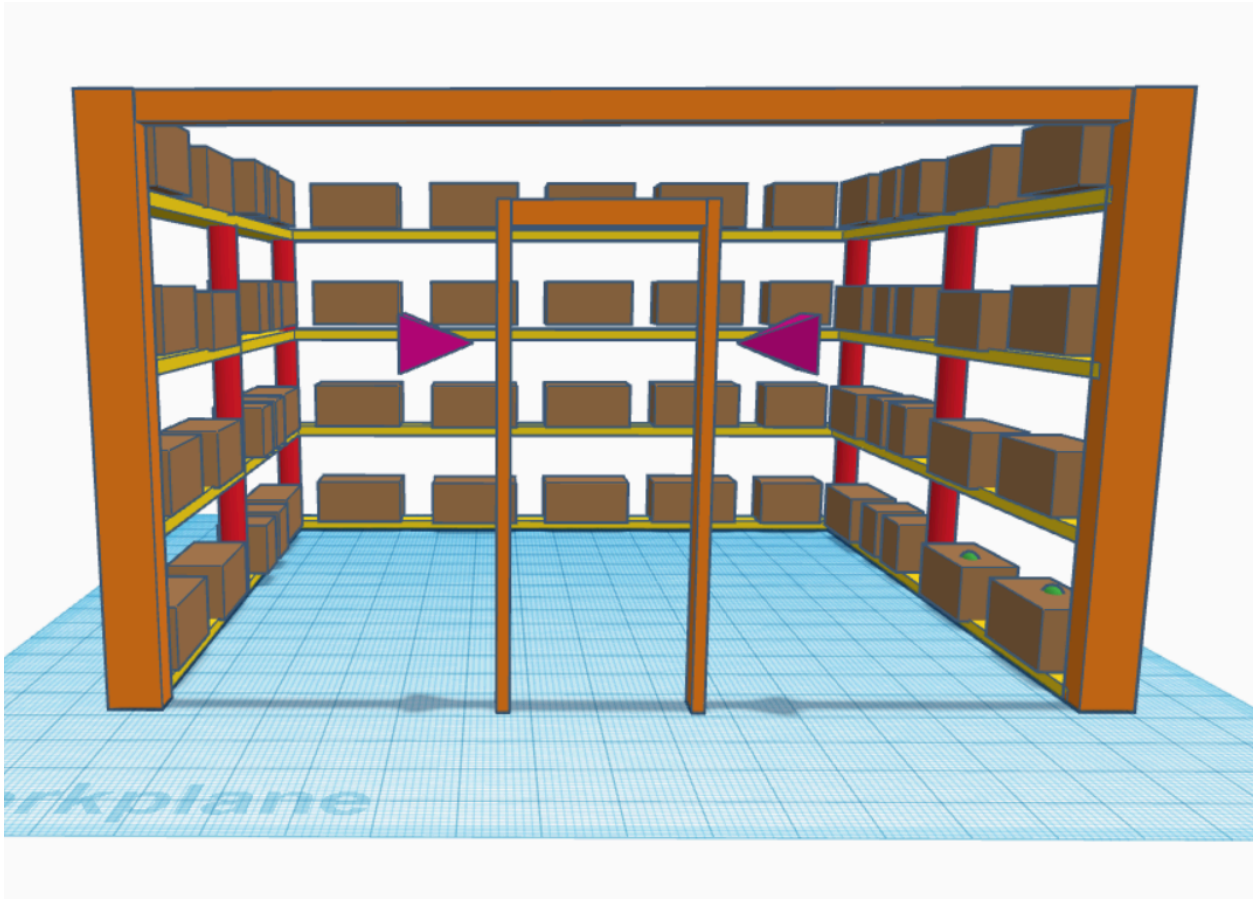
RFID BASED INVENTORY MANAGEMENT SYSTEM.

```
    String jsonData = String("{\"count\":") + count + "}";
    int httpResponseCode = http.PUT(jsonData);

    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Data pushed successfully: " + response);
    } else {
      Serial.println("Error in sending data: " + String(httpResponseCode));
    }

    http.end();
  } else {
    Serial.println("WiFi not connected");
  }
}
```

Now the whole setup is done but where and how we are going to implement it to get the desired output

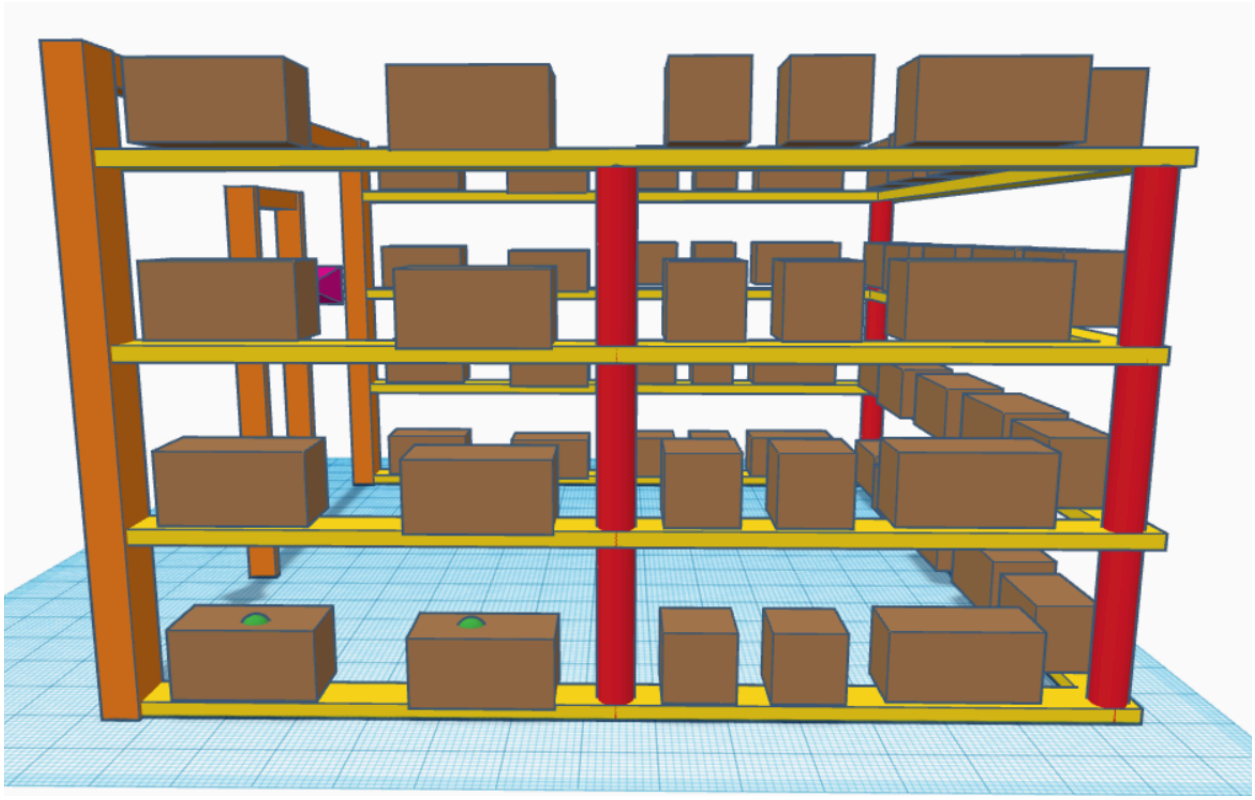
Here is the 3D model for that

Front view:



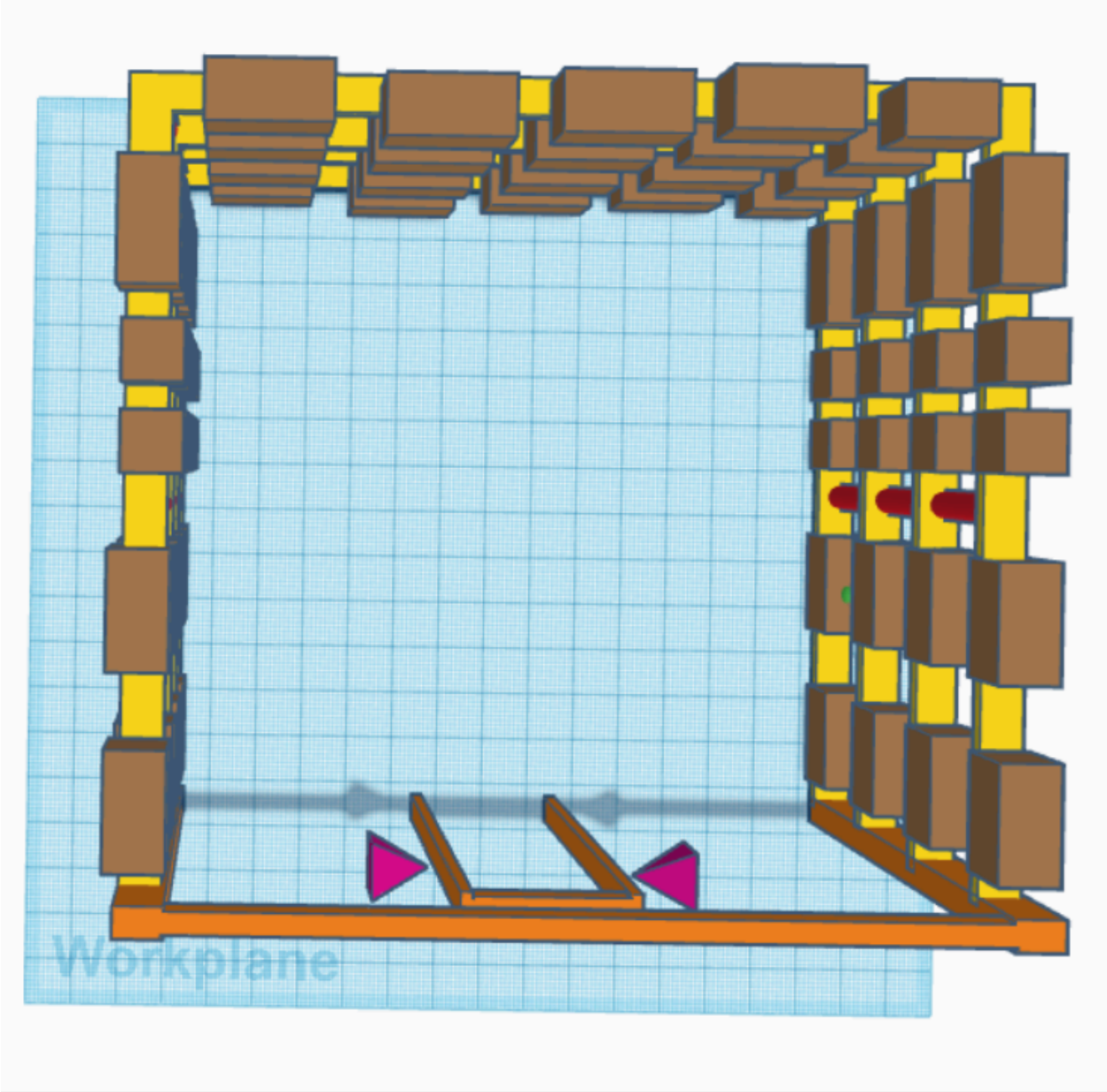
Side view :

RFID BASED INVENTORY MANAGEMENT SYSTEM.

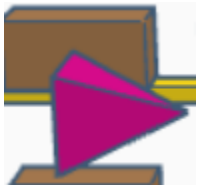


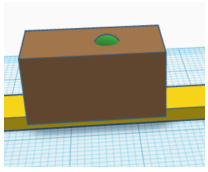
Top view

RFID BASED INVENTORY MANAGEMENT SYSTEM.



Where :


is the mounting of the our module which will detect the passive RFID module


box with RFID tag

RFID BASED INVENTORY MANAGEMENT SYSTEM.

Cad link
https://www.tinkercad.com/things/jMZ0Sl8VKUG-epic-fulffy/edit?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard&sharecode=JfVNsvR38hPKuey03JRZ2ydXLAwpJWs0s4ju11XkZ_g