PROJECT JOYSTICK

CODE DOCUMENTATION

Documentation for the ESP32 code:

main.cpp:

The C++ code is used to control a joystick using an ESP32 and send the values to another device using the at_protocol.

Functionality:

The code reads the values of the joystick and sends them as a string using the at_protocol. The values are in the format "x_value, y_value, b_state" where x_value and y_value are the analog values of the joystick's x and y axes, and b_state is the digital value of the joystick's button.

The string is sent using a put_ak() function of the AtClient library.

AtClient: This is a class that represents an AT client.The AtClient class provides an interface to send and receive AT commands over a serial port.

AtSign: Class that represents an AT sign.The AtSign class provides an interface to work with AT signs.

AtKey: Class that represents an at_key. An at_key is a unique identifier that is used to authenticate a message sent between two at_signs. The AtKey class provides an interface to work with at_keys.

keys_reader: A namespace that contains a function **read_keys** which reads and returns the keys for authentication.

SSID and **PASSWORD**: These are constants that contain the credentials required to connect to the WiFi network.

Libraries:-

<Arduino.h>: The Arduino library which provides basic Arduino functions and macros

<WiFiClientSecure.h>: A library for making secure TCP/IP connections over Wi-Fi with an ESP32

<SPIFFS.h>: A library for accessing files stored on the ESP32 flash file System

<at_client.h>: Contains the definition of a class AtClient, which provides methods to communicate with an at_protocol based server

Functions:-

Setup() - Initializes the pins of the microcontroller board for reading the joystick values and authenticates with the server.

Loop() - Reads the values from the joystick and sends it to the server using the put_ak function of the AtClient class.

analogRead(): This is a function that reads the analog value of a given pin.

digitalRead(): This is a function that reads the digital value of a given pin.

pinMode(): This is a function that sets the mode of a given pin, whether it is an input or output.

delay(): This is a function that pauses the execution of the program for a specified time

Variables:

JOYSTICK_X_PIN: The pin number of the x-axis of the joystick.

JOYSTICK_Y_PIN: The pin number of the y-axis of the joystick.

JOYSTICK_B_PIN: The pin number of the button of the joystick.

at_sign: An object of the AtSign class that represents the sender's at_sign.

java: An object of the AtSign class that represents the receiver's at_sign.

keys: The authentication keys for the at_sign object.

at_client: An object of the AtClient class that handles the connection and communication using the @protocol.

at_key: An object of the AtKey class that represents the message key.

The **put_ak** function is called on the at_client object, passing the at_key and the character array representing the data as arguments.This function is responsible for sending the data to the remote server.

After sending the data, **delay(100)** is called, which pauses the execution of the program for 100 milliseconds before proceeding to the next iteration of the **loop()** function.

This delay can be useful to control the timing of data transmission or to introduce a delay between consecutive data transmissions.

Documentation for the Java App:

App.java

Functionality:

The code receives the joystick values sent by the ESP32 code as a string using the at_protocol.
The values are split into x, y, and b_state and displayed in the text area of the window.
The x and y values are used to update the position of a console label in the window, and the b_state value is used to change the color of the console label.

Libraries:

javax.swing.*: Java's built-in GUI library

java.awt.*: Java's abstract window toolkit

org.atsign.client.api.AtClient: An API for interacting with the at_protocol network

org.atsign.common.*: Common classes used in the at_protocol network

Variables:

outputTextArea: A JTextArea object that displays output in the window.

consolePanel: A JPanel object that displays a console in the window.

consoleLabel: A JLabel object that represents the position of the console in the consolePanel.

consoleX: An integer that represents the x-coordinate of the console in the consolePanel.

consoleY: An integer that represents the y-coordinate of the console in the consolePanel.

consoleColor: A boolean that determines the color of the console.

previousX: An integer that stores the previous x-coordinate of the console in the consolePanel.

previousY: An integer that stores the previous y-coordinate of the console in the consolePanel.

Constructor:

App(): The constructor function that sets up the window and its components This initializes the instance variables and adds components to the window, such as the consolePanel and outputTextArea.

The main method of the **App** class initializes an instance of the class and sets up a connection with a remote server using the **AtClient** and **AtSign** classes from the **org.atsign** package.

It also creates a **SharedKey** object that is used to retrieve data from the remote server.

The method then enters an infinite loop that retrieves data from the shared key, formats it, and passes it to the **displayOutput** method of the **App** class to update the GUI with the joystick data.

The loop is paused for 50 milliseconds at each iteration using **Thread.sleep(50)**

Functions:

1. **displayOutput method:**

The Java code has a method named "displayOutput" that takes a string parameter "output".
The method processes the input data by splitting it into x, y, and z components.
The method updates the position of the console and its color based on the x, y, and z values.
The method appends the output to the outputTextArea.

2. **moveConsole method**:

The Java code has a method named "moveConsole" that takes x and y integers as parameters.
The method updates the position of the consoleLabel in the consolePanel based on the x and y values.
The method scales the x and y values to fit the size of the consolePanel.

3. changeConsoleColor method:

The method named "changeConsoleColor" that takes a z integer as a parameter.
The method updates the color of the consoleLabel based on the z value.

Note:

-The comments in the App.java code provide additional details and explanations for the code.

-You can find a quick explanation video for the code at https://www.youtube.com/watch?v=FQ0Ldu9kGHc