Part 4: Building a basic IoT home security system using a Raspberry Pi and Amazon Web Services

**Introduction:**

Home crime and burglary is an ever-growing worry in less developed or poverty-stricken areas of the world. Home security systems are not cheap nor accessible to most people. This project aims to build a basic IoT security system using a Raspberry Pi and Amazon Web Services (AWS). In the realm of system internals and cyber security, it is important that processing sensitive data is secure from a cloud, software and hardware level. The proposed system will enable visual access into a user home; therefore, must be equipped with best security practices to detour those with malicious intent from accessing the system.

As mentioned previously, AWS will be used to achieve several tasks which would normally require additional hardware. The low running cost and extreme scalability of several AWS services make it an ideal option for this project. [MORE MAYBE]

In order to achieve the aforementioned aim, the following objectives will be attempted:

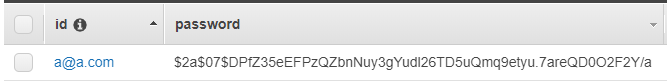
* Create a web interface with a secure login system
* Read a hashed password from AWS DynamoDB to authorise login
* Use the Raspberry Pi camera to take still images and display them to the user on demand.
* Store images securely in an AWS s3 bucket with a timestamp.
* Restrict access to AWS s3 based on IP address
* Utilise system interrupts to implement a ‘panic button’ feature which turns on the alarm.
* Subscribe to a secure MQTT broker to receive requests for initiating the alarm
* Publish to the MQTT broker to turn on the alarm
* Utilise system interrupts to silence the initiated alarm

**Methodology:**

**Create web interface and login system:**

The first step taken to develop the system was to install apache2 on the pi and create the web interface. The front-end consists of a simple layout, utilising Bootstrap for a responsive experience on all size devices. The popular MVC architecture was adopted, keeping database access, controllers and user views separate from each other [1].

The login system was designed using a server-side language – PHP. A default user was manually input into the AWS DynamoDB alongside a hashed password.



References:

[1] <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

[2]

<https://aws.amazon.com/premiumsupport/knowledge-center/block-s3-traffic-vpc-ip/>

THIS WILL GO SOMEWHERE