|  |  |
| --- | --- |
| **(Confidential)** | |
| Scope of disclosure | [Company] for EPH-PJ only |
| Period of confidentiality | 7 year after issued date |
| Head of Information Owner | Head of engineer department |
| Handling restriction | NA |

**Investigation Report**

SCRATCH

**Toshiba Software Development (Vietnam) Co., Ltd**

|  |
| --- |
| Document ID: TSDV-EPH-Scratch-Investigation Report-Scratch |
| Total: 32 Page No. 1 |

**Revision History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rev. No.  (X.YY) | Date (YYYY-MM-DD) | Section No. Changed | Change Description | Author | Reviewed by | Approved by |
| 0.01 | 2021-10-20 | All | Initialize | ThanhBX |  |  |
| 0.02 | 2021-10-27 | 4 | Update detail steps and result | ThaiNH | ThanhBX |  |
| 0.03 | 2021-11-30 | 3 | Add detail design block, scenario | ThaiNH | ThanhBX |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Contents**

[*1.* Introduction 5](#_Toc89769871)

[1.1. Purpose 5](#_Toc89769872)

[1.2. Scope 5](#_Toc89769873)

[1.3. Definitions, Acronyms, Abbreviations 5](#_Toc89769874)

[1.4. References 5](#_Toc89769875)

[*2.* Overall Description 6](#_Toc89769876)

[2.1. Product Perspective 6](#_Toc89769877)

[2.2. Technical Standards 7](#_Toc89769878)

[2.3. User characteristics/Operational Scenarios 7](#_Toc89769879)

[2.4. Operating Environment 7](#_Toc89769880)

[3. Detail design blocks 8](#_Toc89769881)

[3.1. Type definitions 8](#_Toc89769882)

[3.2. List blocks 8](#_Toc89769883)

[3.3. Block detail 9](#_Toc89769884)

[3.3.1. Remote IP Address with port 9](#_Toc89769885)

[3.3.2. Recognize image 9](#_Toc89769886)

[3.3.3. Recognize image on path 9](#_Toc89769887)

[3.3.4. When receive image from EPH 9](#_Toc89769888)

[3.3.5. When receive recognition result from EPH 9](#_Toc89769889)

[3.3.6. Save image 10](#_Toc89769890)

[3.3.7. Show recognition result reporter 11](#_Toc89769891)

[3.3.8. Connect successful 11](#_Toc89769892)

[3.3.9. Disconnect 11](#_Toc89769893)

[3.4. Scenario block samples 12](#_Toc89769894)

[3.4.1. Recognize newly received photo periodically 12](#_Toc89769895)

[3.4.2. Recognize specified image 13](#_Toc89769896)

[3.4.3. Recognize image in network many EPHs 14](#_Toc89769897)

[3.5. Abnormal case 15](#_Toc89769898)

[4. Functional Requirements 16](#_Toc89769899)

[4.1. Add an extension into Scratch 17](#_Toc89769900)

[4.1.1. Prepare development environment 17](#_Toc89769901)

[4.1.2. Implement your block 17](#_Toc89769902)

[4.1.3. Implement GUI 20](#_Toc89769903)

[4.1.4. Confirm extension added 22](#_Toc89769904)

[4.2. Communicate between extension and SDK 24](#_Toc89769905)

[4.2.1. Introduction 24](#_Toc89769906)

[4.2.2. Implement client into extension scratch 24](#_Toc89769907)

[4.2.3. Implement server 27](#_Toc89769908)

[4.2.4. Integrate websocket with SDK 27](#_Toc89769909)

[4.3. Run scenario from Scratch to SDK 29](#_Toc89769910)

[4.3.1. Practice 29](#_Toc89769911)

[4.3.2. Result 30](#_Toc89769912)

[*5.* Appendix 32](#_Toc89769913)

**List of Figures**

[Figure 3. 1 Recognize newly received photo periodically 12](#_Toc89769914)

[Figure 3. 2 Recognize specified image 13](#_Toc89769915)

[Figure 3. 3 Receive result from specified EPH 14](#_Toc89769916)

[Figure 3. 5 Functional overview 16](#_Toc89769917)

[Figure 3. 6 Scratch interface 22](#_Toc89769918)

[Figure 3. 7 Extension EPH/Pi SDK Controller added 23](#_Toc89769919)

[Figure 3. 8 Create example block of extension 24](#_Toc89769920)

[Figure 3. 9 Practice add extension 29](#_Toc89769921)

[Figure 3. 10 Practice add new block of extension 29](#_Toc89769922)

[Figure 3. 11 Communicate log on Raspberry Pi 30](#_Toc89769923)

[Figure 3. 12 Communicate log on EPH 30](#_Toc89769924)

[Figure 3. 13 Communicate log on Scratch 31](#_Toc89769925)

# Introduction

## Purpose

TTEC wants to build a GUI environment for programming, developers can program without writing code in the GUI to interact with the SDK, which is developed by TSDV in term 21A. Scratch was selected as the development.

This document describes:

* How to create new extension and add it to Scratch application.
* Protocol communicates between new extension with SDK.

## Scope

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| ID No | Acronyms | Definition |
| 1 | SDK | Software Development Kit |
| 2 | DDS | Data Distribution Service, a Pub/Sub technology for ubiquitous, polyglot, efficient and secure data sharing |
| 3 | TTEC | Toshiba Tec Corporation |
| 4 | TSDV | Toshiba Software Development Vietnam |
| 5 | API | Application Programming Interface |
| 6 | DCPS | Data-Centric Publish-Subscribe |
| 7 | EPH | Endpoint Hub, a TTEC device which target SDK will be run on |
| 8 | DNN | Deep Neural Network (for image recognition) |

## References

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID No | Reference | Version Number | Document Name/Source & Location reference | Brief Description | Remarks |
| 1 | Requirement | 20210805 | TSDV様\_21A作業のご相談.pdf | Requirement of customer | NA |

# Overall Description

## Product Perspective

Raspberry PI

Scratch

Extensions

Local

communication

SDK

DDS

Network

EPH board

Figure 2. 1 Structure Overview

Investigative subjects in this document are “Extensions” and “Local communication”.

Table 2‑1: Component list

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Function name** | **Description** | **Remark** |
| 1 | Raspberry PI | The Raspberry Pi is a low-cost computer | NA |
| 2 | Scratch | Scratch is a free programming language and online community where you can create your own interactive stories, games, and animations.  https://scratch.mit.edu/ | NA |
| 3 | Extensions | This is an Extension that needs to be added to Scratch so that users can programmatically interact with the SDK | NA |
| 4 | Local communication | Protocol communication between extensions and SDK | NA |
| 5 | SDK | Software Development Kit | NA |
| 6 | DDS network | Data Distribution Service, a Pub/Sub technology for ubiquitous, polyglot, efficient and secure data sharing | NA |
| 7 | EPH board | Endpoint Hub, a TTEC device which target SDK will be run on | NA |

## Technical Standards

* Scratch 3.0
* C++
* JavaScript

## User characteristics/Operational Scenarios

## Operating Environment

* Hardware environment
  + Raspberry Pi 3/4
* Development Environment:
  + OS: Raspbian 10
  + Toolchain:
    - npm: v6.14.15
    - node: v14.18.1
* Testing environment:
  + OS: Raspbian 10
  + Toolchain:
    - npm: v6.14.15
    - node: v14.18.1
* Install NodeJS & npm on Raspberry Pi:
  + sudo apt get install NodeJS npm
  + sudo npm install -g n
  + sudo n 14.18.1

# Detail design blocks

This chapter describes the properties of function blocks and scenario for demonstration, which added to Scratch by TSDV.

## Type definitions

The list of blocks types is shown below:

|  |  |
| --- | --- |
| Block type | Description |
| BOOLEAN | Boolean reporter with hexagonal shape.  Block returns true/false. |
| CONDITIONAL | Specialized command block which may or may not run a child branch.  The thread continues with the next block whether a child branch ran. |
| COMMAND | Execute commands as a function. |
| EVENT | Specialized hat block with no implementation function.  This stack only runs if the corresponding event is emitted by other code. |
| HAT | Hat block which conditionally starts a block stack. |
| LOOP | Specialized command block which may or may not run a child branch.  If a child branch runs, the thread evaluates the loop block again. |
| REPORTER | General reporter with numeric or string value. |

Table 3. Type block definitions

## List blocks

TSDV proposes several blocks for communication between Raspberry Pi & EPH.

|  |  |  |
| --- | --- | --- |
| **No** | **Name Block** | **Type** |
| 1 | Remote IP Address with port | COMMAND |
| 2 | Recognize image | COMMAND |
| 3 | Recognize image on path | COMMAND |
| 4 | When receive image from EPH | HAT |
| 5 | When receive recognition result from EPH | HAT |
| 6 | Save image | COMMAND |
| 7 | Show recognition result | REPORTER |
| 8 | Connect successful | BOOLEAN |
| 9 | Disconnect | COMMAND |

Table 3. Overview blocks

For details on what each block means, please refer to the next section.

## Block detail

### Remote IP Address with port

|  |  |  |
| --- | --- | --- |
| **Name:** | Remote IP Address [Ip address] with port [port] | |
| **Type:** | COMMAND | |
| **Arguments:** | **Name** | **Meaning** |
|  | Ip address | Ip address want to connect. |
| Port | Port connects to WebSocket server opened. |
| **Description:** | This setting specifies connection with WebSocket server. | |
| **Interface:** |  | |

Table 3. Remote IP Address with port

### Recognize image

|  |  |  |
| --- | --- | --- |
| **Name:** | Recognize image | |
| **Type:** | COMMAND | |
| **Arguments:** | **Name** | **Meaning** |
|  | N/A | N/A |
| **Description:** | Send request recognition newly image received from specified EPH. | |
| **Interface:** |  | |

Table 3. Recognize image latest

### Recognize image on path

|  |  |  |
| --- | --- | --- |
| **Name:** | Recognize image latest on [Path] | |
| **Type:** | COMMAND | |
| **Arguments:** | **Name** | **Meaning** |
|  | Path | Image paths want to recognize. |
| **Description:** | Send request recognition image with a clear name. | |
| **Interface:** |  | |

Table 3. Recognize image on path

### When receive image from EPH

|  |  |  |
| --- | --- | --- |
| **Name:** | When receive image from [EPH] | |
| **Type:** | HAT | |
| **Arguments:** | **Name** | **Meaning** |
|  | EPH | Select specified EPH capture image. |
| **Description:** | Get trigger capture image from specified EPH. | |
| **Interface:** |  | |

Table 3. Get last capture image on EPH

### When receive recognition result from EPH

|  |  |  |
| --- | --- | --- |
| **Name:** | When receive recognition result from [EPH] | |
| **Type:** | HAT | |
| **Arguments:** | **Name** | **Meaning** |
|  | EPH | Select specified EPH recognize image. |
| **Description:** | Get trigger recognition result from specified EPH. | |
| **Interface:** |  | |

Table 3. Get last recognize text on EPH

### Save image

|  |  |  |
| --- | --- | --- |
| **Name:** | Save image | |
| **Type:** | COMMAND | |
| **Arguments:** | **Name** | **Meaning** |
|  | N/A | N/A |
| **Description:** | Save image received from EPH | |
| **Interface:** |  | |

Table 3. Save image

### Show recognition result

|  |  |  |
| --- | --- | --- |
| **Name:** | Show recognition result | |
| **Type:** | REPORTER | |
| **Arguments:** | **Name** | **Meaning** |
|  | N/A | N/A |
| **Description:** | Show recognition result via reporter | |
| **Interface:** |  | |

Table 3. Show recognition result via reporter

### Connect successful

|  |  |  |
| --- | --- | --- |
| **Name:** | Connect successful | |
| **Type:** | BOOLEAN | |
| **Arguments:** | **Name** | **Meaning** |
|  | N/A | N/A |
| **Return** | True | Connected successful |
|  | False | Not connected |
| **Description:** | Check if connected successful to WebSocket server. | |
| **Interface:** |  | |

Table 3. Is not connected?

### Disconnect

|  |  |  |
| --- | --- | --- |
| **Name:** | Disconnect | |
| **Type:** | COMMAND | |
| **Arguments:** | **Name** | **Meaning** |
|  | N/A | N/A |
| **Description:** | Disconnect to WebSocket server. | |
| **Interface:** |  | |

Table 3. Disconnect

## Scenario block samples

TSDV proposes some scenarios describe relationship between blocks.

### Recognize newly received photo periodically

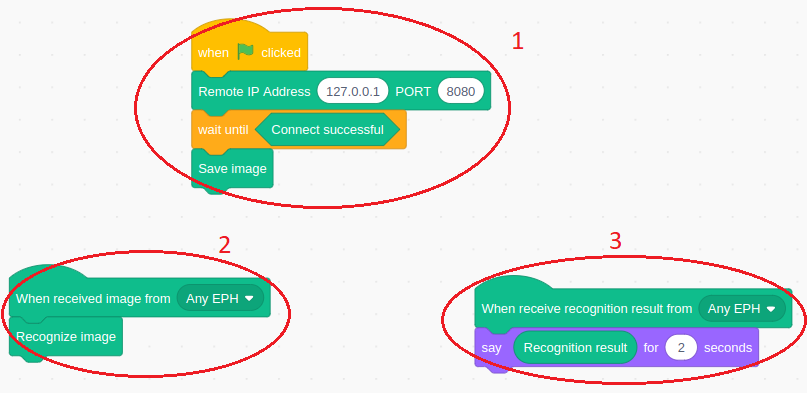


Figure 3. Recognize newly received photo periodically

**Sequence 1:**

* Scratch connects to WebSocket server

**Sequence 2:**

* Pi starts receiving image from EPH
* Each time a new image arrives (indicate by "When received image" block), Pi saves that image locally ("Save image" block)
* Pi requests the recognition result for the received image using "Recognize last image" block

**Sequence 3:**

* Every time a recognition result arrives, Scratch displays that result.

### Recognize specified image

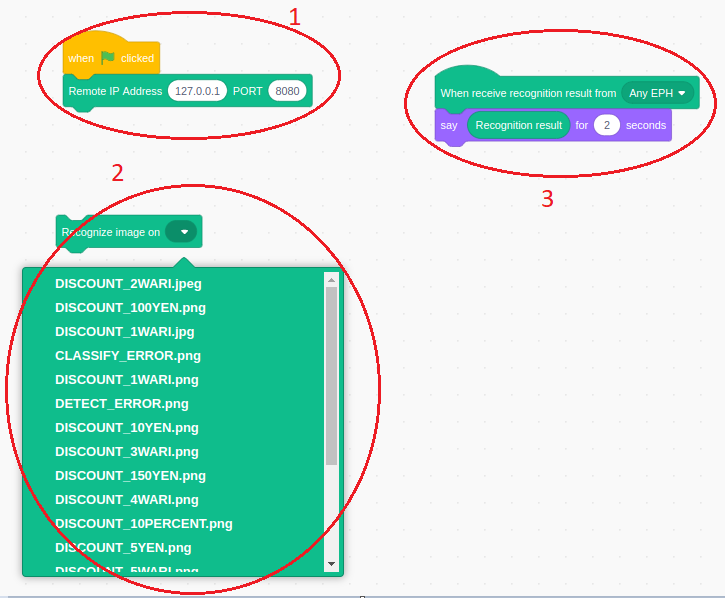


Figure 3. Recognize specified image

**Sequence 1:**

* Scratch connects to WebSocket server

**Sequence 2:**

* User selects an image saved locally on Pi and requests recognition result for that image

**Sequence 3:**

* Scratch displays the recognition result

### Recognize image in network many EPHs

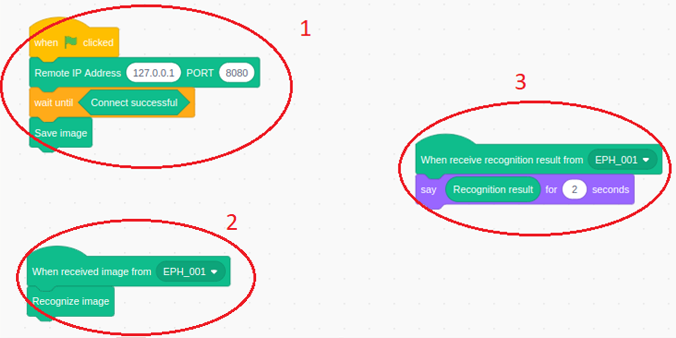


Figure 3. Receive result from specified EPH

**Sequence 1:**

* Scratch connects to WebSocket server

**Sequence 2:**

* Pi request recognition result for each received image

**Sequence 3:**

* Scratch displays recognition result from a specific EPH

## Abnormal case

### Connect to WebSocket server failure

Display log message when user enters incorrect Ip Address or Port.

Log message:

**“Connection error. Please check the remote IP address or your network connection”.**

### Select EPH not match between 2 HAT blocks

Refer section 3.4.3

Example: If user selects EPH\_001 in block “When receive image from [EPH]” but selects EPH\_002 in block “When receive recognition result from [EPH]” => no display recognition result.

### Request recognize empty

Refer section 3.4.2

If user don’t select specified image after that recognize image => display log message “**Not found image recognize**”.

# Functional Requirements

This section describes how to create new extension & communicate extension Scratch and SDK.

1. Add new extension into Scratch
2. Communicate between extension and Scratch
3. Run example scenario from Scratch to SDK

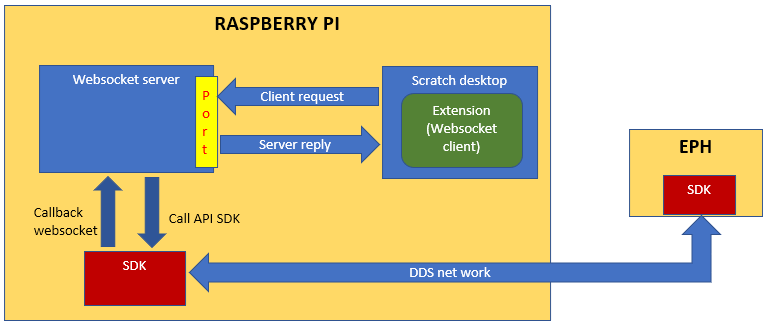


Figure 3. Functional overview

## Add an extension into Scratch

### Prepare development environment

* Download clones of 3 git repositories (scratch-vm, scratch-gui, scratch-desktop)

$ git clone https://github.com/LLK/scratch-vm.git

$ git clone <https://github.com/LLK/scratch-gui.git>

$ git clone <https://github.com/LLK/scratch-desktop.git>

* Checkout branch newest build scratch desktop required

Open /scratch-desktop/package.json -> find branch scratch-gui required.

You will see: **"scratch-gui": "github:LLK/scratch-gui#scratch-desktop-v3.27.0",**

$ cd scratch-gui

$ git checkout scratch-desktop-v3.27.0

Open /scratch-gui/package.json -> find branch scratch-vm required.

You will see: **"scratch-vm": "0.2.0-prerelease.20211103090905",**

$ cd scratch-vm

$ git checkout 0.2.0-prerelease.20211103090905

### Implement your block

We create a new extension called “EPH/Pi SDK Controller”. Each extension can have multiple blocks.

Add (★) and update (★★) some files.

/scratch-vm

|  |  |
| --- | --- |
|  | ├── LICENSE |
|  | ├── node\_modules |
|  | ├── package.json |
|  | ├── package-lock.json |
|  | ├── README.md |
|  | ├── src |
|  | │   ├── blocks |
|  | │   ├── dispatch |
|  | │   ├── engine |
|  | │   ├── extensions |
|  | │   │   ├── scratch3\_ev3 |
|  | │   │   ├── scratch3\_pi\_eph\_sdk 【★ ADDED】 |
|  | │   │   │   └── index.js【★ ADDED File 1】 |
|  | │   │   ├── 【OTHERS】 |
|  | │   │   └── scratch3\_wedo2 |
|  | │   ├── extension-support |
|  | │   │   ├── argument-type.js |
|  | │   │   ├── extension-manager.js　【★★ MODIFIED File 2】 |
|  | │   │   └── 【OTHERS】 |
|  | │   ├── import |
|  | │   ├── index.js |
|  | │   ├── util |
|  | │   └── virtual-machine.js |
|  | ├── test |
|  | ├── TRADEMARK |
|  | ├── webpack.config.js |
|  |  |
|  |  |

(★) In File 1 **src/extension/scratch3\_newblocks/index.js**, add a block interface and process.



(★★) File 2 **src/extension-support/extension-manager.js** is for an extension menu implementation.

In this file, add **scratch3\_newblocks defined** in previous File 1.



### Implement GUI

Add some lines at the last of **src/lib/libraries/extensions/index.jsx** file in the same directory **scratch-gui**

* Import path image icon, image background (★)
* Add source code GUI (★★)



* Add background image as name **newBlockImage.png** in the **newBlock** directory.
* Add menu icon image as name **newBlockButtonImage.png** in the **newBlock** directory.

/scratch-gui

|  |
| --- |
|  |
|  | ├── LICENSE |
|  | ├── package.json |
|  | ├── prune-gh-pages.sh |
|  | ├── README.md |
|  | ├── src |
|  | │   ├── components |
|  | │   ├── containers |
|  | │   ├── css |
|  | │   ├── examples |
|  | │   ├── index.js |
|  | │   ├── lib |
|  | │   │   ├── alerts |
|  | │   │   ├── libraries |
|  | │   │   │   └── extensions/ |
|  | │   │   │   │   ├── index.jsx　【★　MODIFIED】 |
|  | │   │   │   │   ├── 【OTHERS】  | | | | ├── newBlock 【★　ADDED】 |
|  | | | | | | └── newBlockButtonImage.png 【★　ADDED】 |
|  | | | | | | ├── newBlockImage.png.png 【★　ADDED】 |
|  | │   │   ├── log.js |
|  | │   ├── playground |
|  | │   ├── reducers |
|  | │   └── test.js |
|  | ├── static |
|  | ├── test |
|  | ├── TRADEMARK |
|  | ├── translations |
|  | ├── webpack.config.js |
|  | ├── npm-error.log |
|  | └── npm.lock |

### Confirm extension added

$ cd scratch-vm

$ npm install

$ npm run build

$ cd scratch-gui

$ npm install ../scratch-vm

$ npm run build

$ npm link

$ cd scratch-desktop

$ npm install

$ npm link scratch-gui

$ npm run build:dist

After building, scratch desktop will store : **/dist/Scratch 3-3.26.0-arm64.AppImage**

Execute: ./**Scratch 3-3.26.0-arm64.AppImage**

**Result**:

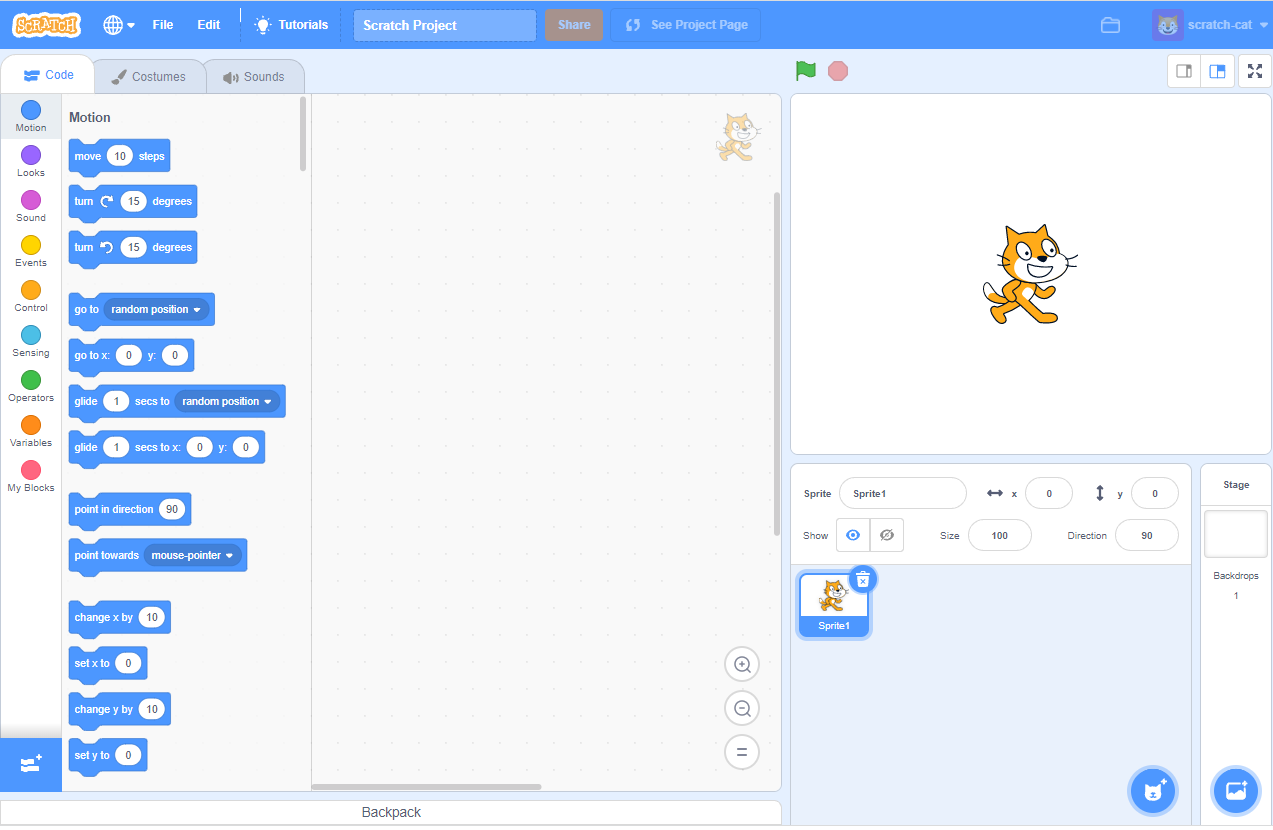


Figure 3. Scratch interface

“Add extension” button

Click “Add extension” button:

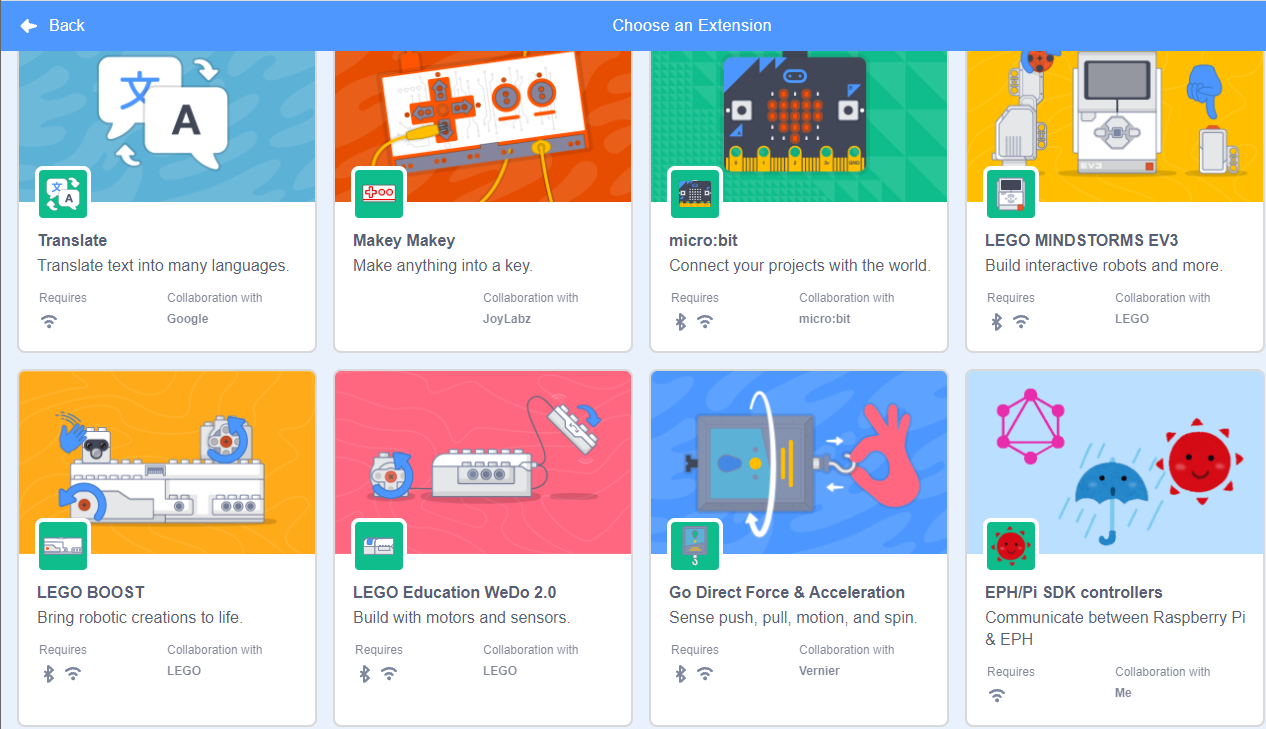


Figure 3. Extension EPH/Pi SDK Controller added

* **EPH/Pi SDK controller** extension is created successful.

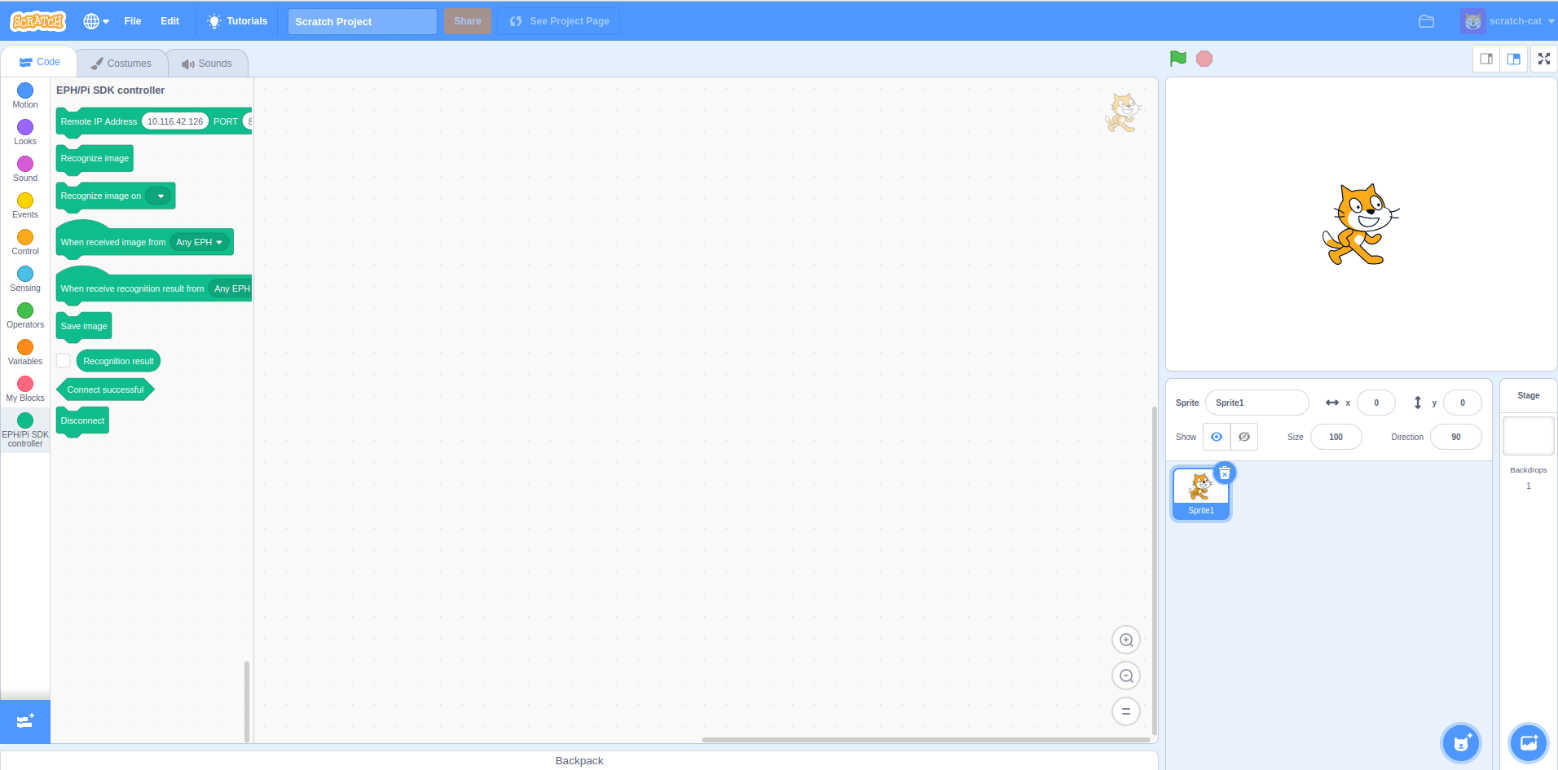


Figure 3. Create example block of extension

## Communicate between extension and SDK

### Introduction

Use WebSocket to connect extension Scratch & SDK.

WebSocket enables bidirectional communication over the server/client.

WebSocket can be run together with a normal HTTP server. You can click a block in scratch desktop application, and your Raspberry Pi which request recognize image to EPH or save image from EPH with communication going both ways.

### Implement client into extension scratch

/scratch-vm

|  |  |
| --- | --- |
|  | ├── LICENSE |
|  | ├── node\_modules |
|  | ├── package.json |
|  | ├── package-lock.json |
|  | ├── README.md |
|  | ├── src |
|  | │   ├── blocks |
|  | │   ├── dispatch |
|  | │   ├── engine |
|  | │   ├── extensions |
|  | │   │   ├── scratch3\_ev3 |
|  | │   │   ├── scratch3\_newblocks |
|  | │   │   │   └── index.js【★ MODIFIED】 |
|  | │   │   ├── 【OTHERS】 |
|  | │   │   └── scratch3\_wedo2 |
|  | │   ├── extension-support |
|  | │   │   ├── argument-type.js |
|  | │   │   ├── extension-manager.js |
|  | │   │   └── 【OTHERS】 |
|  | │   ├── import |
|  | │   ├── index.js |
|  | │   ├── util |
|  | │   └── virtual-machine.js |
|  | ├── test |
|  | ├── TRADEMARK |
|  | ├── webpack.config.js |
|  |  |
|  |  |

We add source code of client (JavaScript) to **index.js (**★)

Please refer source code at section 4.1.2.

### Implement server

We implement a C++ server using the WebSocket++ WebSocket library. It is provided here so that it might be used as the basis for communicate with Scratch.

Reference: <https://github.com/adamrehn/websocket-server-demo>

Dependencies:

* WebSocket++
* Asio
* JsonCpp

License:

* WebSocket++ - BSD 3-Clause License (see server/external/LICENSE/websocketpp.txt)
* Asio - Boost Software License (see server/external/LICENSE/asio.txt)
* JsonCpp - Public Domain / MIT License (see server/external/LICENSE/jsoncpp.txt)

### Integrate WebSocket with SDK

/webserver

|  |  |
| --- | --- |
|  | ├── external |
|  | ├── CMakeList.txt |
|  | ├── WebsocketServer.cpp |
|  | ├── WebsocketServer.h |
|  | ├── main.cpp【★ ADD】  ├── WebsocketSupporter.cpp【★★ ADD】  ├── WebsocketSupporter.h【★★ ADD】 |

We add source code of server (C++) to **main.cpp (**★)



## Run scenario from Scratch to SDK

**NOTE**: Currently scratch desktop application not yet show recognize result which sent from SDK. So, we’re testing with scratch web version.

### Practice

* Start scratch
  + $ cd scratch-gui
  + $ npm start
  + After building successful, go to <http://localhost:8601/> - the playground outputs the default GUI component
* Click button “Add extension” and select “EPH/Pi SDK controller”

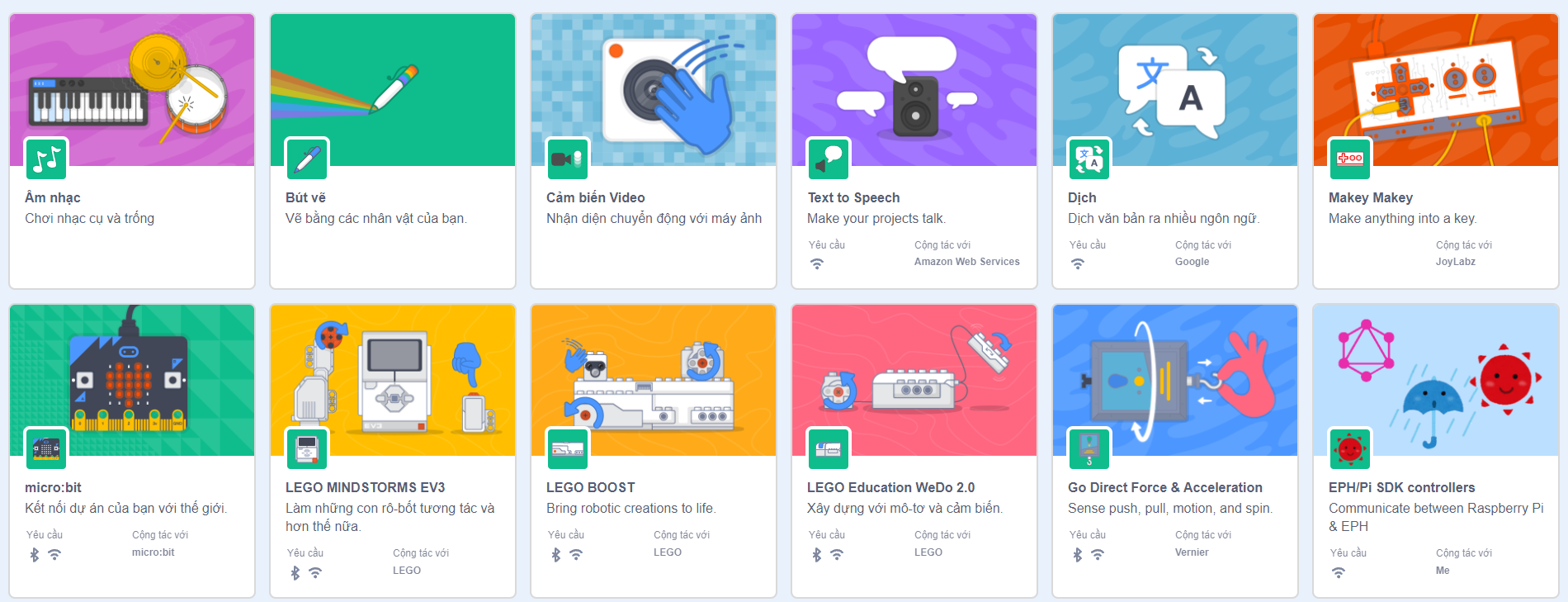


Figure 3. Practice add extension

* Choose a block from **EPH/Pi SDK Controller** extension. Example: Select block “**Recognize image on**”
* Edit “path” into image path which want to recognize
* Example: ./DISCOUNT\_2WARI

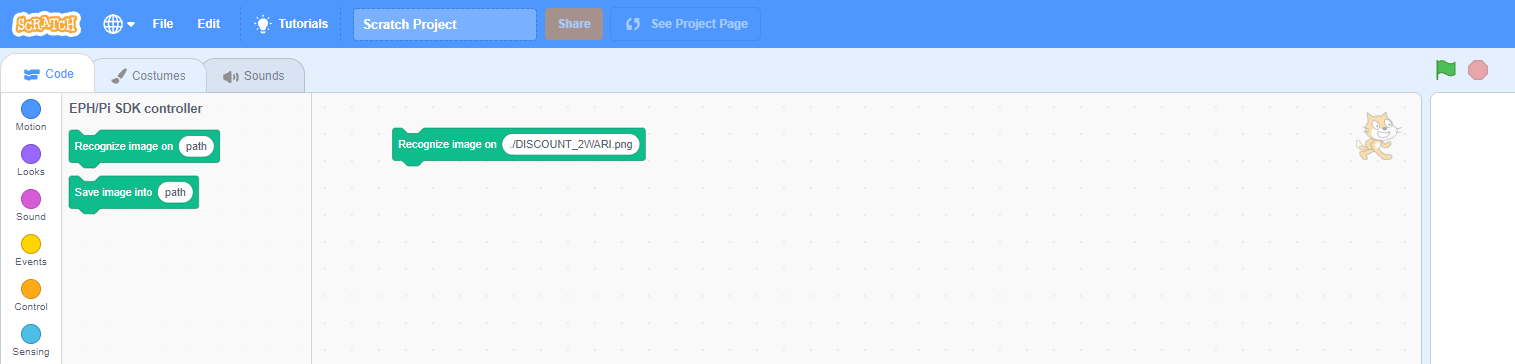


Figure 3. Practice adds new block of extension

* Start webserver
* Start EPH recognize Image Job
* Start DNN Job
* Click “Go” to start request recognize image

### Result

We try run scenario as section 3.4.1 and result is below:

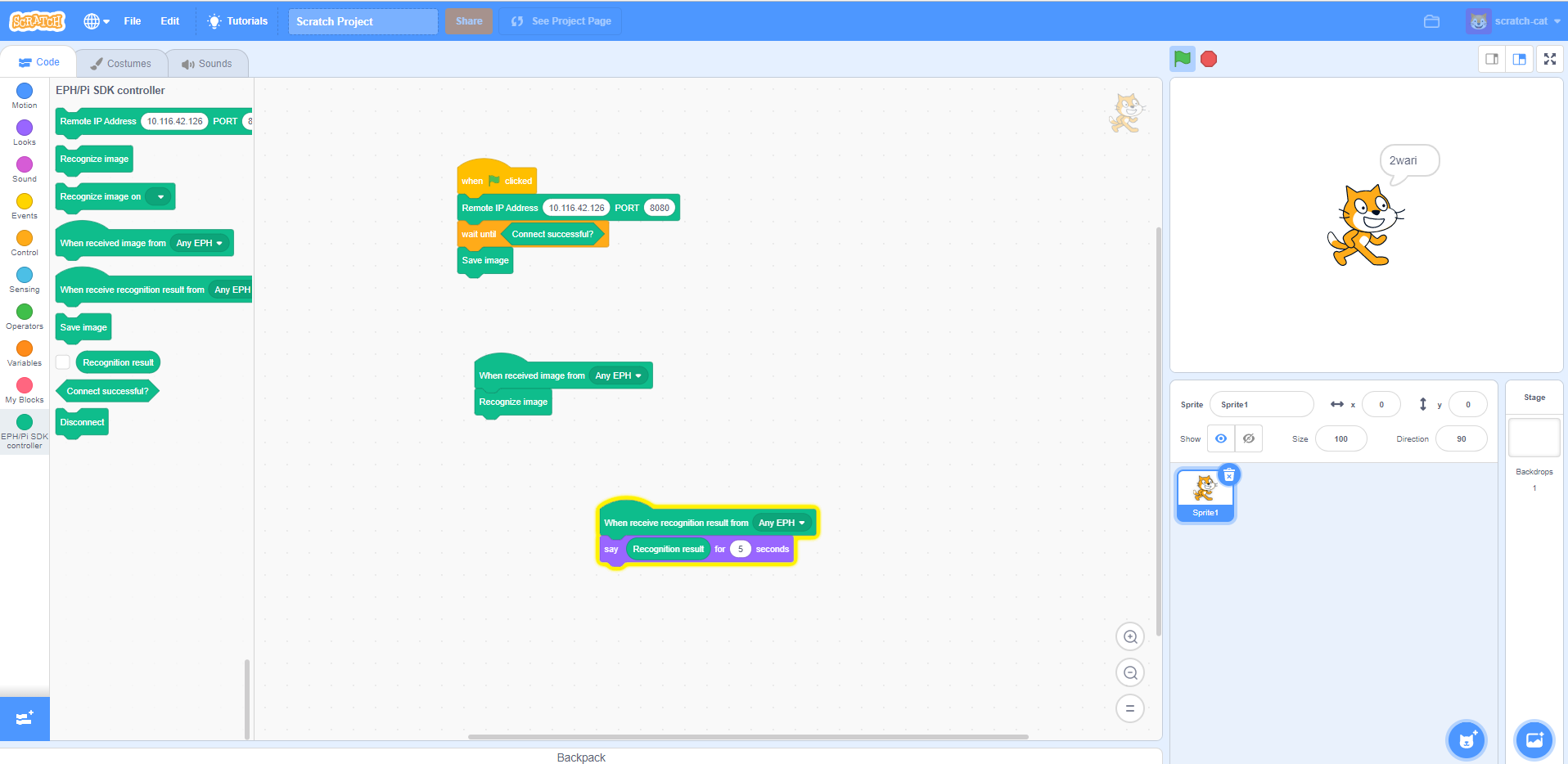


Figure 4. Recognize successful

# Appendix

*NA*