|  |  |
| --- | --- |
| **(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 |

**User Manual**

SCRATCH

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

|  |
| --- |
| Document ID: TSDV-21B-EPH-UserManual-Scratch |
| Total: 13 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 | 2021-12-24 | All | Initialize | ThaiNH | ThanhBX |  |
| 1.00 | 2021-12-30 | 3 | Add user manual | ThaiNH | ThanhBX |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Contents**

[*1.* Introduction 5](#_Toc91783335)

[1.1. Purpose 5](#_Toc91783336)

[1.2. Scope 5](#_Toc91783337)

[1.3. Definitions, Acronyms, Abbreviations 5](#_Toc91783338)

[1.4. References 5](#_Toc91783339)

[*2.* Overall Description 6](#_Toc91783340)

[2.1. Product Perspective 6](#_Toc91783341)

[2.2. Technical Standards 7](#_Toc91783342)

[2.3. User characteristics/Operational Scenarios 7](#_Toc91783343)

[2.4. Operating Environment 7](#_Toc91783344)

[3. User Manual 8](#_Toc91783345)

[3.1. Components 8](#_Toc91783346)

[3.2. How to build 8](#_Toc91783347)

[3.2.1. Build Scratch on Raspberry Pi 8](#_Toc91783348)

[3.2.2. Build SDK on EPH 8](#_Toc91783349)

[3.2.3. Build websocket server on Raspberry Pi 8](#_Toc91783350)

[3.3. How to use 9](#_Toc91783351)

[3.3.1. Start Scratch desktop 9](#_Toc91783352)

[3.3.2. Scenario 1: Recognize newly received photo periodically 9](#_Toc91783353)

[3.3.3. Scenario 2: Recognize specified image 10](#_Toc91783354)

[3.3.4. Scenario 3: Recognize image in network many EPHs 11](#_Toc91783355)

[*4.* Appendix 13](#_Toc91783356)

**List of Figures**

[Figure 3. 1 Recognize newly received photo periodically 9](#_Toc91783357)

[Figure 3. 2 Recognize specified image 10](#_Toc91783358)

[Figure 3. 3 Receive result from specified EPH 11](#_Toc91783359)

# 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 build, use Scratch desktop.
* How to use scenario block samples

## 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 | Investigate report | 0.03 | TSDV-EPH-Scratch-Investigation Report-Scratch.docx | Investigate report | NA |

# Overall Description

## Product Perspective

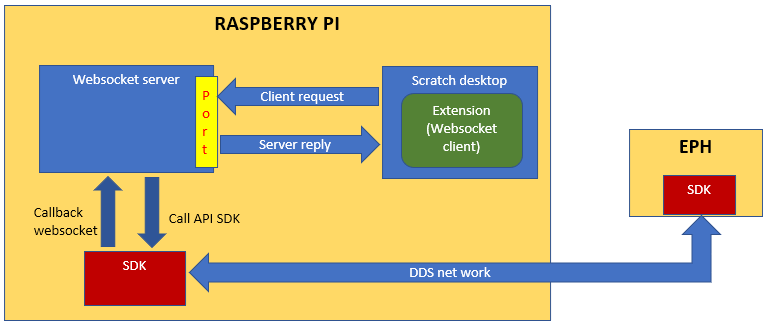


Figure 2. Functional Overview

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
  + 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

# User Manual

This chapter describes the ways how to use Scratch Desktop application with some scenarios.

## Components

**NOTE**: Because the websocket server is a separate part from the Scratch desktop, in order the server to start at the same time as Scratch the server path will be relatively dependent on the Scratch path.

So if you want to run Scratch somewhere please copy all folder **Scratch** as below.

**Release structure folder**

\**Scratch** (For Raspberry Pi)

|--- Scratch.AppImage (Desktop application)

|--- run.sh (Script start websocket server)

|--- extension (Source code add extension scratch)

| websocket-server

| |--- build (Build folder)

| |--- images (images template folder)

| |--- install (Install folder)

| |--- sdk (Source code SDK)

| |--- src (Source code websocket server)

…

**\SDK** (For EPH)

|--- sdk (Source code SDK)

…

## How to build

### Build Scratch on Raspberry Pi

TSDV provided Scratch application.

In case want to rebuild, please refer section 3.6 in **TSDV-21B-EPH-Investigation-Report-Scratch.docx**.

### Build SDK on EPH

Require:

* CMake
* OpenSplice V6\_9\_190925OSS
* OpenCV

Build:

* $ source /opt/opensplice/HDE/armv8.linux/release.com (depend on your device)
* $ cd sdk/app && mkdir build
* $ cd build
* $ cmake .. -DCMAKE\_INSTALL\_PREFIX=<installation folder>
* $ cmake –-build .
* $ cmake -P cmake\_install.cmake [-DBUILD\_TYPE=<Debug|Release>]

### Build websocket server on Raspberry Pi

Require:

* CMake
* OpenSplice V6\_9\_190925OSS
* OpenCV

Build:

* $ source /opt/opensplice/HDE/armv8.linux/release.com (depend on your device)
* $ cd websocket-server && mkdir build
* $ cd build
* $ cmake .. -DCMAKE\_INSTALL\_PREFIX=<installation folder>
* $ cmake –-build .
* $ cmake -P cmake\_install.cmake [-DBUILD\_TYPE=<Debug|Release>]

**NOTE:** In project, TSDV using <installation folder> = ../install

In case you want to use other <installation folder>, please modify **run.sh** in release folder.

## How to use

### Start Scratch desktop

* Double click to Scratch app image application.
* Choose extension EPH/Pi SDK Controller

### Scenario 1: Recognize newly received photo periodically

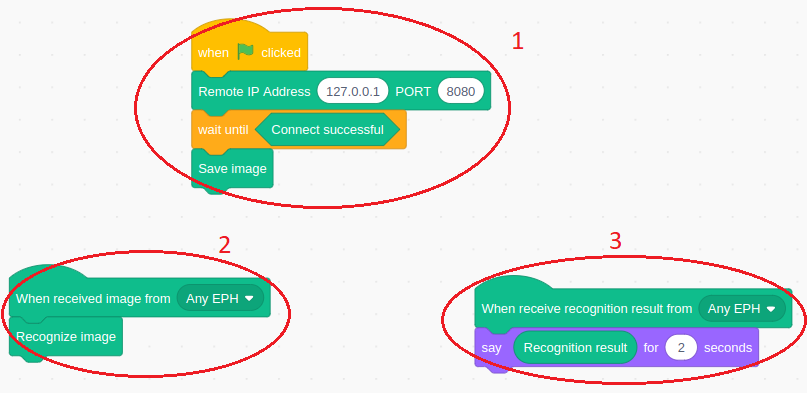


Figure 3. Recognize newly received photo periodically

**Step 1**: Start job **capture image** and **recognize image** on EPH board.

* Open new terminal and run DNN application

$ ./StickerRecognitionStreamingDummyApp

* Open new terminal and run capture image job (Currently EPH have not yet captured image, TSDV send sample an image to Raspberry Pi like as EPH captures the image). (Please put **sample.jpg** into <installation folder>)

$ source [path opensplice installed]/release.com

$ cd <installation folder>/bin

$ ./ephlib\_sample capture\_image

* Open new terminal and run recognize image job

$ source [path opensplice installed]/release.com

$ cd <installation folder>/bin

$ ./ephlib\_sample recognize\_image

**Step 2***:* Use block as figure 3.1

* Click to icon “Go”  to start scenario.
* User can use block “Disconnect” to disconnect to websocket server and shutdown all jobs.

### Scenario 2: Recognize specified image

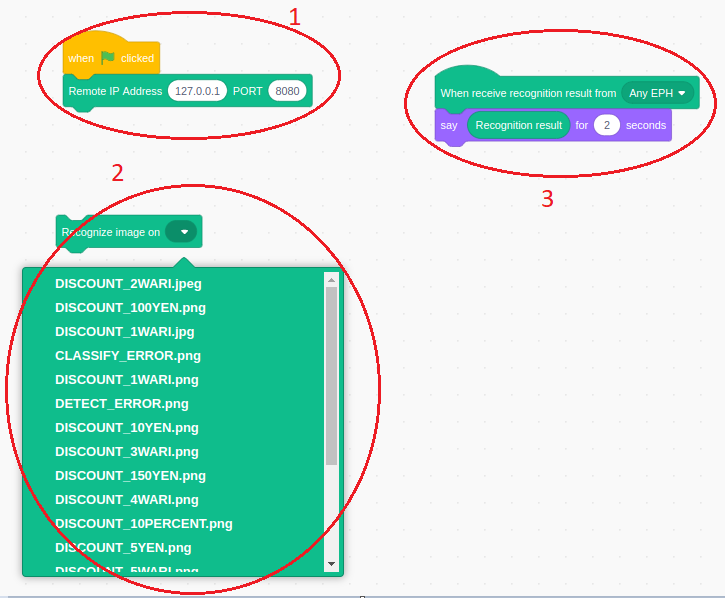


Figure 3. Recognize specified image

**Step 1**: Start job **recognize image** on EPH board

* Open new terminal and run DNN application

$ ./StickerRecognitionStreamingDummyApp

* Open new terminal and run recognize image job

$ source [path opensplice installed]/release.com

$ cd <installation folder>/bin

$ ./ephlib\_sample recognize\_image

**Step 2***:* Use block as figure 3.2

* Click to icon “Go”  to connect to web socket server.
* Select any image appear on list.
* Click to block “Recognize image on” to recognize image specified.
* User can use block “Disconnect” to disconnect to websocket server and shutdown all jobs.

**NOTE**: After click “Go”, list image in block “Recognize image on” appears. User can select any image in list to recognize.

The image list is created when an image is received from the EPH or user input.

Path: websocket-server/install/bin/images

### Scenario 3: Recognize image in network many EPHs

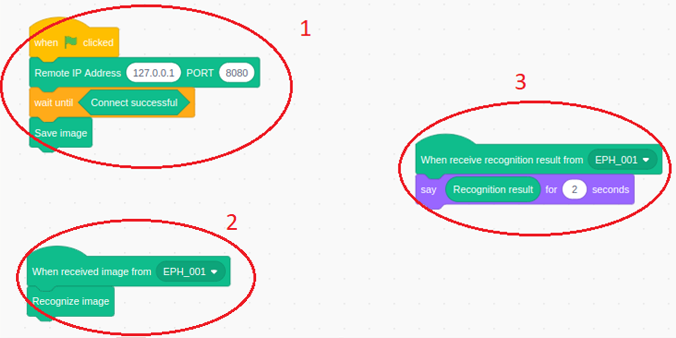


Figure 3. Receive result from specified EPH

**Step 1***:* Start job **capture image** and **recognize image** on EPH board.

* Open new terminal and run DNN application

$ ./StickerRecognitionStreamingDummyApp

* Open new terminal and run capture image job (Currently EPH have not yet captured image, TSDV send sample an image to Raspberry Pi like as EPH captures the image). (Please put **sample.jpg** into <installation folder>)

$ source [path opensplice installed]/release.com

$ cd <installation folder>/bin

$ ./ephlib\_sample capture\_image

* Open new terminal and run recognize image job

$ source [path opensplice installed]/release.com

$ cd <installation folder>/bin

$ ./ephlib\_sample recognize\_image

**Step 2***:* Use block as figure 3.3

* Click to icon “Go”  to start scenario.
* User can select specified EPH in runtime.
* User can use block “Disconnect” to disconnect to websocket server and shutdown all jobs.

**NOTE**: Currently TSDV just develop processing for 2 EPH. At 2nd EPH, please modify source code SDK for ***deviceID*** and rebuild: (For build SDK please refer section 3.1.2)

* sdk/app/dummy\_app/c/lib/c/ ephlib\_recognizeimage.cpp
* sdk/app/dummy\_app/c/lib/c/ ephlib\_capture.cpp

Before: static const std::string deviceId = "EPH\_001";

Modify: static const std::string deviceId = "EPH\_002";

After build SDK on 2nd EPH, please follow **Step 1** and **Step 2** as 1st EPH.

# Appendix

*NA*