The LightSwarm Project

"Raspberry Pi IoT Projects: Prototyping Experiments for Makers”

by John C. Shovic, Apress, 2016

Self-Organizing Systems

- Self-organization is defined as a process where some sort of order arises out of the local interactions between smaller items in an initially disordered system.
- There is no central control of who is the master and who is the slave. This makes the system more reliable and able to function even in a bad environment.
- Typically these kinds of systems are robust and able to survive in a chaotic environment. Self-organizing systems occur in a variety of physical, biological, and social systems.
- One reason to build these kinds of systems is that the individual devices can be small and not very smart, and yet the overall task or picture of the data being collected and processed can be amazingly interesting and informative.

LightSwarm

- The major design specifications for the LightSwarm software are the following:
  - Device self discovery.
  - Device becomes master when it has the brightest light; all others become slaves.
  - Distributed voting method for determining master status.
  - Self-organizing swarm. No server required.
  - Swarm must survive and recover from devices coming in and out of the network.
  - Master device sends data to Raspberry Pi for analysis and distribution to Internet.

Code Structure

- There are two major modules written for LightSwarm.

1. The first is ESP8266 code for the LightSwarm device itself (written in the Arduino IDE simplified C and C++ language), Listings 2-2 though 2-11, or https://github.com/switchdoclabs/LightSwarm/blob/master/LightSwarm.ino
   - TCS34725 code is in https://github.com/adafruit/Adafruit_TCS34725
   - https://github.com/adafruit/Adafruit_TCS34725/blob/master/Adafruit_TCS34725.h
2. The second is the data-gathering software (written in Python on the Raspberry Pi).

Swarm Commands

The system uses the broadcast mode of UDP so when LightSwarm devices send out a message to the WiFi subnet, everybody gets it if they are listening on port 2910, then they can react to it.

- DEFINE_SERVER_LOGGER_PACKET - This is the new IP address of the Raspberry Pi so the LightSwarm device can send data packets;
- LIGHT_UPDATE_PACKET - Packet containing current light from a LightSwarm device. Used to determine who is master and who is slave;
- BLINK_BRIGHT_LED - Command to a LightSwarm device to blink the bright LED on the TCS34725.
- LOG_TO_SERVER_PACKET - Packets sent from LightSwarm devices to Raspberry Pi;
- RESET_SWARM_PACKET - All LightSwarm devices are told to reset their software;
- RESET_ME_PACKET - Just reset a particular LightSwarm device ID;
- MASTER_CHANGE_PACKET - Packet sent from LightSwarm device when it becomes a master (not implemented);
- CHANGE_TEST_PACKET - Designed to change the master / slave criteria (not implemented);

Swarm Table (on Every Node)

```
int swarmClear[SWARMSIZE]; // 0
int swarmVersion[SWARMSIZE];
int swarmState[SWARMSIZE];
long swarmTimeStamp[SWARMSIZE]; // for aging, -1
int swarmAddresses[SWARMSIZE]; // Swarm addresses, 0
```

Packet Info

byte packetBuffer[BUFFERSIZE]; //buffer to hold incoming and outgoing packets, PACKET_SIZE = 14, BUFFERSIZE = 1024;

packetBuffer[0] = 0xF0; // StartByte

packetBuffer[1] == command type

packetBuffer[2] == swarmID

packetBuffer[3] == state, 1 for master, 0 for slave

packetBuffer[4] == version

packetBuffer[5] * 256 + packetBuffer[6] == clear

packetBuffer[7] * 256 + packetBuffer[8] == R

packetBuffer[9] * 256 + packetBuffer[10] == G

packetBuffer[11] * 256 + packetBuffer[12] == B

Code Flow

Setup():

1. Print all local info to the serial window
2. Flash onboard LED every 500ms
3. Setup Swarm ID
    1. mySwarmID is the array index, initialized to 0
    2. Connect to network, retry in a loop of 500ms
    3. Set up the UDP port and initialize swarm data array
    4. Set SwarmID using the IP address

Code Flow

Loop(): *pages 31-43*

- read all the data from the TCS34725 sensor to find out how bright the ambient light currently is
- check if any incoming UDP packet broadcast to port 2910.
- record info and check for a Master
  1. setAndReturnMySwarmIndex(int incomingID)@page 40: check for new ID index in the array
  2. checkAndSetIfMaster()@page 38-40: check timestamp of any data in the array, and decide I Am Master
  3. all packet commands are checked for proper actions

Code Flow

Loop(): *pages 36-37*

- The step before the last step in loop() is to broadcast an update packet
- broadcastARandomUpdatePacket()@page 37:
  - Delay randomly
  - Set the last octet of the IP address in the following function to 255, which is the UDP broadcast address

- sendLightUpdatePacket(IPAddress & address)@page 36: send out light packets to the swarm.

Code Flow

Loop(): *pages 41-43*

- The last step in the loop() is to send log to server
- sendLogToServer() @page 41: send log packet to the server if I am master and server address defined
  - packet created as a string with the following format:
  - swarmID, MasterSlave, SoftwareVersion, clearColor, Status | .... next Swarm ID 0,1,15,3883, PR | 1,0,14,399, PR | ....

Sample System Output

- Page 59 shows a message dump from a device
- Page 60 shows a message dump from a logger

Logging Software on RPi

- Read and log information on the swarm behavior.
- Reproduce archival swarm behavior.
- Provide methods for testing swarm behavior (such as resetting the swarm).
- Provide real-time information to the Internet on swarm behavior and status.

- It used the RasPiConnect control panel software, but not available any more.

  https://github.com/milocreek/RasPiConnectServer

  http://www.switchdoc.com/wp-content/uploads/2014/07/RasPiConnectTutorial.pdf

Logger Commands

- The code is designed for a data refresh or push a button on the RasPiConnect control panel. The RasPiConnect server sends a command to the LightSwarm logging software that is running on a thread in the system

1. processCommand(s) - When a command is received from the RasPiConnect server software, this function defines all the actions to be completed.
2. completeCommandWithValue(value) - call function and return a value to RasPiConnect when you completed a command.
3. completeCommand() - call function when you completed a command to tell RasPiConnect you are done.