Camera uCAM-III and IBM Watson visual recognition

Smart systems, HAMK

Bc. David Pivovar
pivovar.david@gmail.com

Assignment

1. Get picture from camera module connected to Arduino MKR1000
2. Send image data to IBM IoT platform
3. Process image by IBM Watson visual recognition v3

Solution

Arduino gets picture captured by camera uCAM-III. Then image is send to IBM IoT platform using MQTT protocol. There it is modified in Node-Red and classified by IBM Watson visual recognition.

Components:

- Arduino MKR1000 (https://store.arduino.cc/arduino-mkr1000-wifi)
- 4D Systems uCAM-III (https://4dsystems.com.au/mwdownloads/download/link/id/420/)

IBM Cloud:

- Internet of Things platform (https://cloud.ibm.com/catalog/services/internet-of-things-platform)
- Node-RED application (https://cloud.ibm.com/developer/appservice/starter-kits/59c9d5bd-4d31-3611-897a-f94eea80dc9f/nodered)
- Visual recognition (https://cloud.ibm.com/developer/watson/starter-kits/bc39151b-3ec2-3e7b-b754-2473883922e9/visual-recognition-nodejs-app)

Solution is at github page https://github.com/pivovard/Arduino/tree/master/CAM/CAM

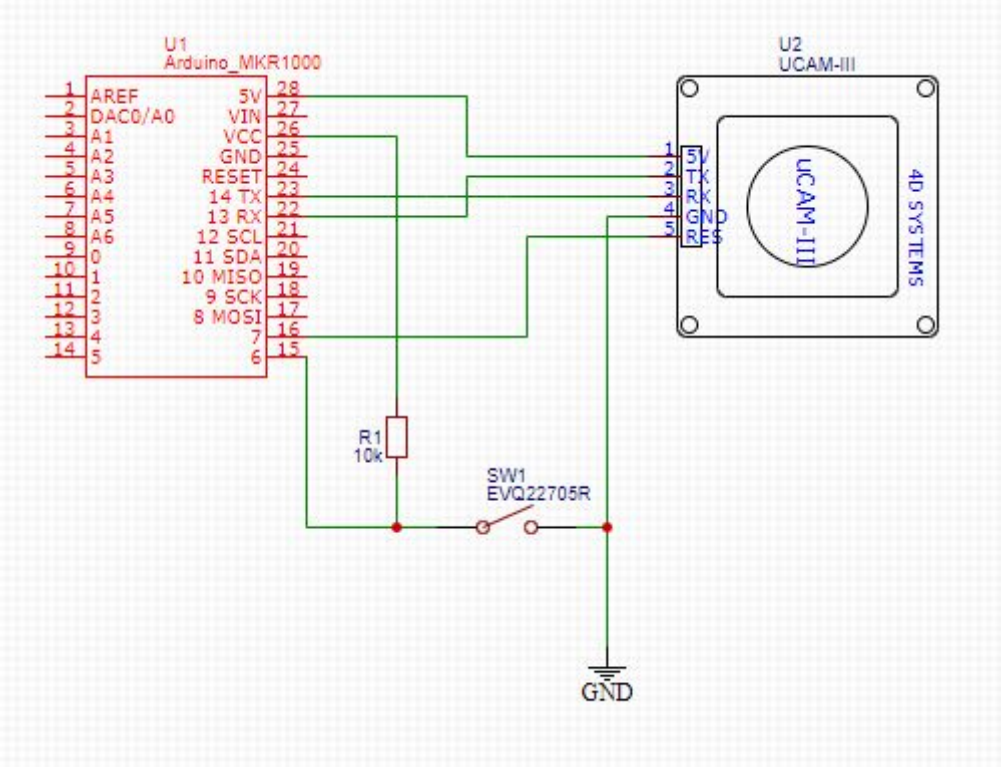

Figure 1: Connection scheme

4D Systems uCAM-III

Camera communicate with serial interface UART. Camera can take pictures in RAW or JPEG format. Resolution of JPEG format can be up to 640x480.
This camera has got one big lack that focus must be made manually on the module.
I created custom library for communication with camera.

Command set

Each command is made of 6 bytes. First byte is always 0xAA followed by one byte command. Next four bytes are parameters. Command set is at picture below.

Figure 2: Command set

Communication

First we must initialize the camera. This is made with SYNC command. We send SYNC command until camera answers with ACK command and handshake is made.
Than we can set parameters. Every setting command is confirmed by ACK/NAK.
We receive the image by sending GET_PICTURE command with format type as parameter. Camera sends size of the image and starts sending data in packages of chosen size (default 64B or 512B). We confirm every package by sending ACK command. First 2 bytes in package are package ID, next 2 bytes are data size in package. Than data follows and last 2 bytes are verifying code. Data are stored in byte array (uint8_t).

Communication with IBM IoT platform

Arduino communicate with IoT platform with MQTT protocol. After acquiring image from camera it sends JSON event about size of image and starts sending data. Library used for MQTT protocol allows to send maximum 128B data. This is very limiting and because images are large data, they need to be send in multiple packages.
First I was sending data as binary event, size of package was 64B (there is a long string as topic header). The issue is that when there is 0 in the data IoT platform discard it and doesn’t

receive rest of the message. Because data are byte type, which is unsigned 0-255, I couldn't substitute zeros by other value.
Only option left is to send data in JSON. That cuts the size of message to the half (32B) because of added JSON load such as commas and brackets.
Next problem was with data size. IoT platform was able to receive only certain amount of data (or packages). When data reached over this size (around 3kB) platform doesn't receive whole image. That's the reason I had to lower image resolution to 160x128 and with high contrast images that wasn't enough.
In my opinion other communication protocols, such as TCP/IP, UDP/IP or HTML, would be much more better and more efficient to send large image data.

Image processing and recognition

On the IoT platform side data are processed. First data must be completed from the packages. Data are stored in flow context variable array. After complete data are received they are converted to the Node.js Buffer. Next the data are send to the visual recognition node to classify. Only general clasifier is used.
To show the image I use **node-red-contrib-image-output** (https://flows.nodered.org/node/node-red-contrib-image-output).

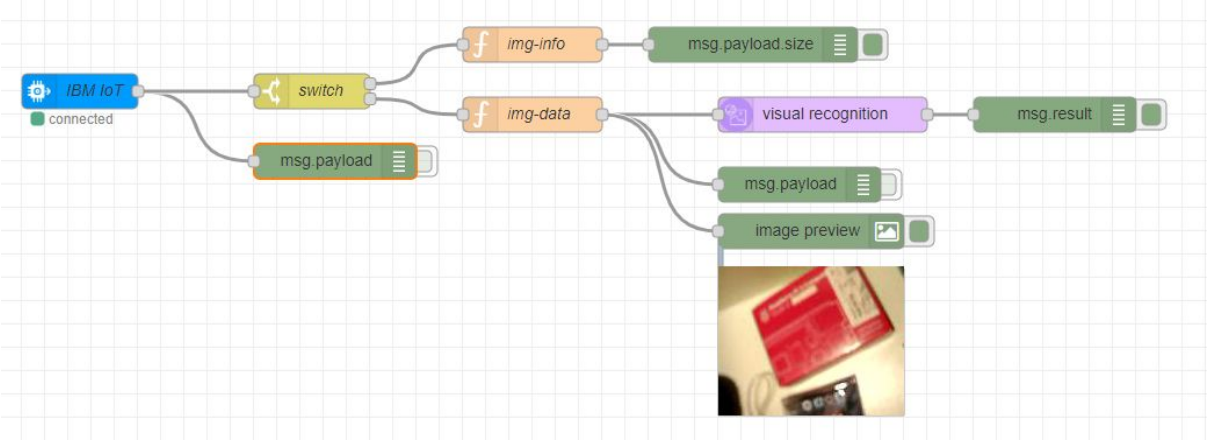

Figure 3: Node-Red flow

Conclusion

I managed to acquire image from uCAM-III camera and I created a library that can be used in the future as well.
The biggest issues I experienced were with communication with IBM IoT platform. It is working but it allows to send only small packages and total size of the data isn't large. Because of this we can send images in low resolution only. This leads to worse classifying as well. Small packages cause longer load time. Other communication protocols that allows to send large data would suit better.
For classification I successfully used IBM Watson visual recognition. Because of lack of time I used only basic classifier that works very well. Even with low resolution fuzzy image it recognized most significant objects with 0.75 accuracy.