USB Power Meter Reading - Documentation

Saroj Panda

July 9, 2021

1 System Architecture

The System Architecture is schematically illustrated in Fig. 1. The IoT device under observation is powered through an USB Power meter, UM25C [5], which provides a Windows Bluetooth application to display the instantaneous information sent from the power meter. The Power meter monitors the power drawn by the IoT device, computes the energy consumption every half a second and sends that information to the Bluetooth application, which then displays that in the text field **Energy(mWh)** as shown in the **image** within the ***Smart Energy Meter*** application in Fig. 1. The background application, ***Smart Energy Meter***, in our framework extracts the energy information from the display image to compute the energy consumption.

1.1 Software Components

The **computer vision** based framework for computing energy consumption consists of two software components namely ***Smart Energy Meter***, and the ***Energy Interface*** as shown in Fig. 1.

1. ***Energy Interface:*** This interface runs on the IoT device as part of the entity/application that needs to measure the device energy consumption. It provides a persistent TCP/IP connection to the Windows machine running the ***Smart Energy Meter*** application and two commands to be sent to that application. To measure the energy for any experiment we will send the ***Start*** command before starting the experiment and the ***Stop*** command at the end of the experiment to receive the energy consumption measurement.

2. ***Smart Energy Meter:*** This is the heart of the framework. This background application developed in Python, captures the computer screen that includes UM25C Bluetooth application window displaying the energy information from the UM25C power meter upon receiving the commands ***Start*** and ***Stop*** from the IoT device. It uses the Python package **PyAutoGUI** [6], which allows any python script to control the mouse

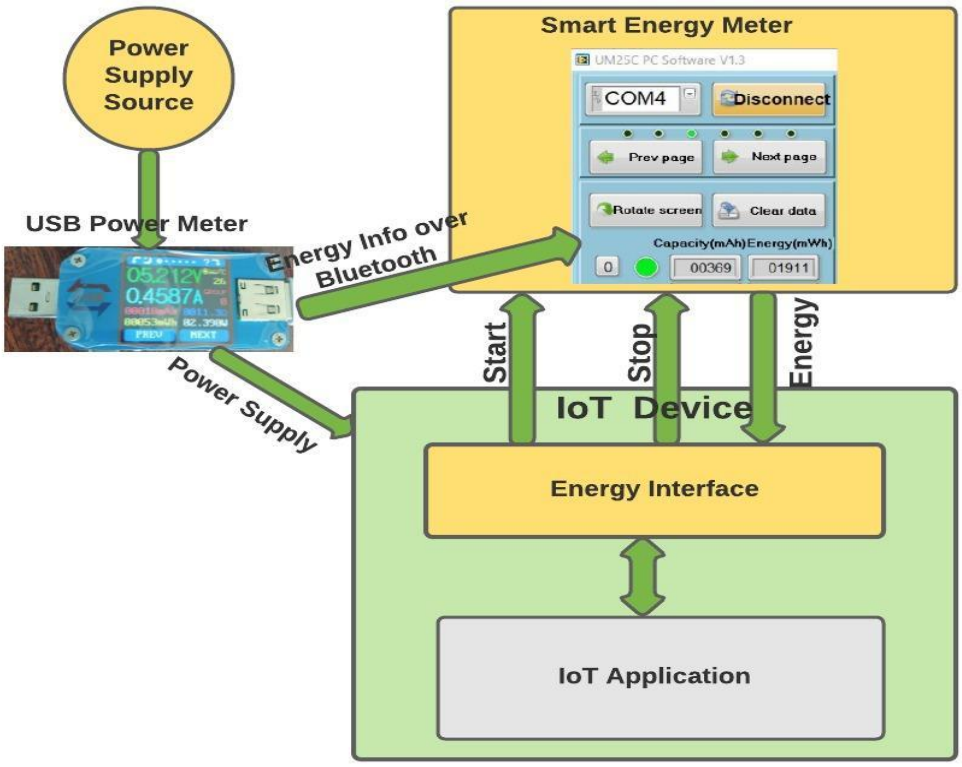

Figure 1: System Architecture of IoT Device Energy Measurement

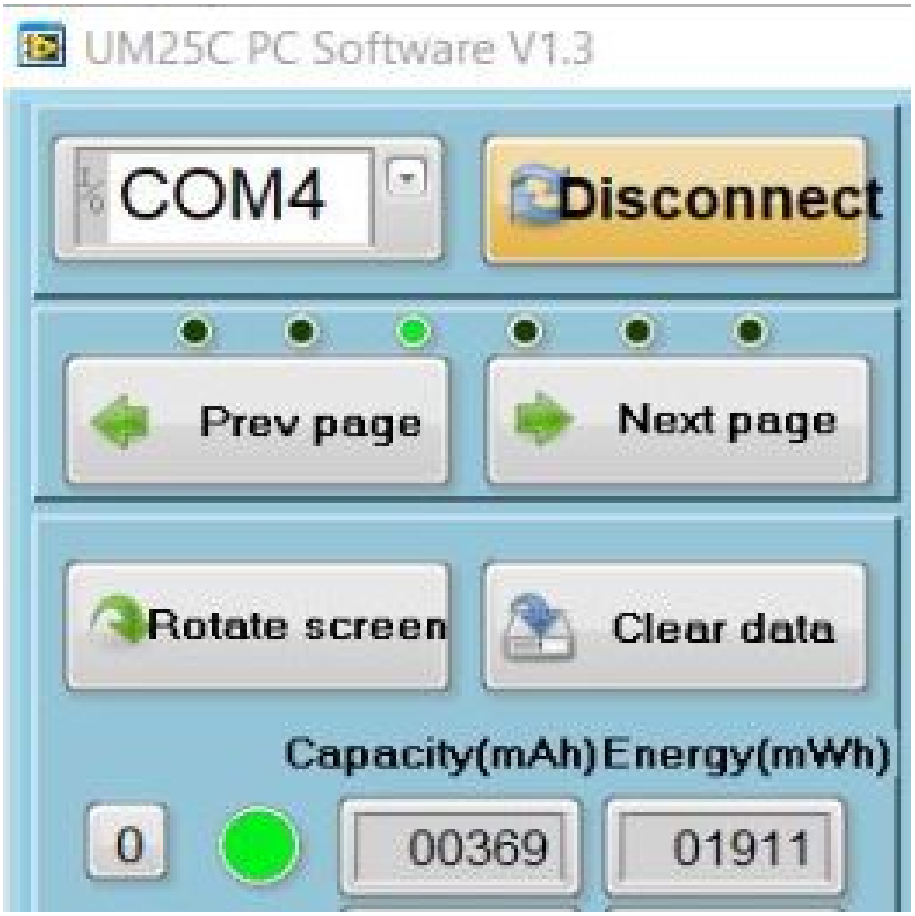

Figure 2: UM25C USB Power Meter Energy Field

and keyboard to automate interactions with other applications including capturing the screen.

After the screens are captured, it uses **computer vision** algorithms using **OpenCV-Python** [4] to process the images in order to locate the energy fields i.e. the text field **Energy(mWh)** as shown in Fig. 2. **OpenCV-Python** is a library of Python bindings to the OpenCV library designed to solve computer vision problems. We make use of the green circle present very close to the text field **Energy(mWh)** as shown in Fig. 2 and store that as a separate image. The **OpenCV-Python matchTemplate** library function performs a template match of the green circle image on the captured screen images to identify the center coordinates of the green circle. As the UM25C Bluetooth application window is a fixed size window, a fixed size pixel navigation in the XY coordinates from the center of the green circle gives access to the whole text field **Energy(mWh)**. Using those navigation pixels, the application extracts the energy fields from those images as separate images.

The extracted images are then processed through the **OpenCV-Python computer vision** algorithms to scale up the image size, binarize (black-and-white) the image, and remove noise from them. The processed images are then passed to Tesseract [3], an optical character recognition (OCR) engine to extract the digits and hence the energy readings. The difference between the two readings (***Start*** and ***Stop***) is returned as the energy consumption of the experiment to the IoT device. We use Tesseract 4 with a new OCR engine, which uses a **neural network** system based on **Long short-term memory (LSTM)** models and has higher accuracy than the legacy OCR engine. The new OCR engine has in-built trained **LSTM** models for many languages and also provides options to train a new model from scratch or fine-tune an existing model. For our framework, the trained **LSTM** model for the English language suffices our requirement.

After all the energy consumption readings are obtained for all the experiments, the requesting IoT application sends the command ***Exit*** to the ***Smart Energy Meter*** application and closes the connection. Similarly, the ***Smart Energy Meter*** application on receiving the command ***Exit*** closes the connection from it's end and waits for next energy consumption measurement request.

1.2 Hardware Components

The energy consumption measurement framework requires a hardware device such as UM25C [5] USB Power Meter with Bluetooth application support. The UM25C can be powered on by supplying power through a charger that has power output 5.0 - 5.1V and 3-3.5A. In the absence of a charger, the UM25C USB Power Meter can be powered through the USB ports of Desktops or Laptops.

2 Evaluation

We used the framework to measure energy consumption of two devices, Raspberry Pi 4 Model B [2] and Nvidia Jetson Nano 2GB [1] that are used as IoT devices for many IoT applications. We used MNIST Image classification application as the test application and run the experiment multiple times. The energy measurement framework accurately returned the energy consumption for each experiment consistent with the previous results for each of those devices.

References

[1] Nvidia Corporation. *Jetson Nano 2GB Developer Kit.* https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit.

[2] Raspberry Pi Foundation. *Raspberry Pi 4 Model B Specifications.* https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[3] Google. *Tesseract, An optical character recognition (OCR) engine.* https://opensource.google/projects/tesseract.

[4] Olli-Pekka Heinisuo. *A library of Python bindings designed to solve computer vision problems.* https://pypi.org/project/opencv-python/.

[5] Glen liu. *UM25C Documents and Software.* https://www.mediafire.com/folder/q2b8h079hpywq/UM25.

[6] Al Sweigart. *A Python Package to automate interactions with other applications.* https://pyautogui.readthedocs.io/en/latest/.