1. Automation in OPV Characterization
   1. Pika Demultiplexer
      1. Concept
      2. Circuit Design and 3D printing
      3. Software and operation
   2. EQE Measuring Setup
      1. Concept and Components
      2. XZ Stage
      3. Filter Flipper
      4. Software and operation
         1. Developed Libraries
         2. Main Software

3 Automation in OPV Characterization

ABSTRACT

This chapter is focused on the development of experimental setups that accelerate, systematize and automatize the measurement of a variety of optoelectronic devices within our group. Since this subject is of great interest for a majority of our group members, mainly because it affects the quality of their work as well as greatly reduces the time and effort needed to perform measurements, we decided to dedicate an entire chapter to give a bit more insight into the experimental setups that make this progress possible. We will focus on two pieces of equipment which build up on each other, a solar cell automated demultiplexer and an automatic EQE measuring setup which also uses the previously mentioned demultiplexer. They are both designed to be easy to assemble, reliable and low cost, so that they can be implemented easily on different environments. To achieve such goals they use commercial PCB manufacturing processes, which are nowadays extremely affordable and widely available, and of the shelf components widely used on the 3D printing industry which provide for cheap, easily replaceable parts which can be sourced from a variety of suppliers. Furthermore, all the pieces used to assemble them a part from the ones previously mentioned are 3d printed so that anyone with a really inexpensive 3D printer can entirely replicate these setups without any complicated machining or injection moulding process.

3.1 Pika IV Characterization Setup

This is the most used setup within this work as well as by other members of the group thanks to its simplicity and reliability. On its core it is composed by three main pieces, the Demultiplexer Board, and the IV Characterization Software and a source meter, a Keythley 2400 in our case. Aside from the spectrometer, both the Demultiplexer Board and the IV Characterization software have been fully developed within this work and have undergone a process of continuous optimization through several iterations throughout the entirety of my Doctorate.

3.1.1 Concept

The main problem this setup is trying to solve arises from the fact that we, as a group, are focused on the manufacturing of medium scale optoelectronic devices, with a strong focus on solar cells. To do so we rely on scalable techniques such as Dr Blade which are capable of covering respectable areas. In that process we generate samples which sometimes exhibit a continuous electrode but more often than not are purposely separated into smaller samples for a variety of reasons, such as compositional studies, annealing temperature gradients, thickness gradients, horizontal tandems… The nature of these discontinuous electrode allows us to sample our devices at several points, and thus assess their performance along the sample, which, depending on the type of study that we are performing, will provide a wide variety of results. This is a very interesting concept, since it allows for an extremely fast optimization while minimizing the amount of material and the total number of samples but it also comes with some problems. Perhaps one of the main issues of this approach is the fact that there are, for our particular samples, 24 electrically separated devices to be connected to the source meter, for each sample. On top of that, on an average solar cell batch we use anywhere from 4 to 12 different substrates, that need most if not all of their individual devices characterized thoroughly. We can see that individually measuring 288 discrete solar cell “pixels” for what has only been under 3 days of device manufacturing time can easily add up to the total time spent by the researcher between experiment planning and results. To further worsen the situation, this raw data needs to be processed file by file, further adding to the overall time expenditure.

The main goal of the Pika IV Characterization Setup is to drastically shorten the characterization time so that the researcher can focus on other, more important tasks, and to provide more reliable results at a much faster pace. To achieve this goal, we have attacked the two main aspects of the problem, the physical and the informatic.

In order to perform the electrical connections in a fast and reliable way we have designed a printed circuit board (PCB) that accommodates a 3d printed holder with gold plated pogo pins, which provide a reliable connection to each and every of the 24 devices present on each substrate as well as multiple ground connections. These pins are connected to a series of low power bistable relays that reroute the connections down to a single connection effectively demultiplexing the signal. In this point there is two BNC connectors that carry the signal to any source meter to be measured. Within the PCB there are a number of buttons that switch the relays as well as an Arduino Nano that is used to automatize relay actuation, as well as a variety of other features which will be further explained later. When we combine this with any computer we can automatically select any of the 24 devices just by sending a serial command. That is one of the most powerful advantages of this setup with respect to others because it can be coupled with any programming language as long as it provides some basic serial communication tools, which the vast majority of programming languages do. Which brings us to the software aspect.

In order to alleviate the burden of measuring each and every single of the 24 devices within each cell, we have written a piece of software using the graphic programming language LabView NXG which provided us with tools to develop a user-friendly Graphical User Interface, as well as powerful backend processing to perform all the measurements, data processing and communication with all the devices. This software application takes care of the automatic detection and connection of all the necessary equipment, the data acquisition and processing, and all the plotting file organization so that the user can leave the measuring station with all the necessary results. These results consist on all the JV Curve files saved in a series of txt files; an excel sheet with all the processed data; an OriginLab file with both all the JV curve data, their individual plots and a collective plot, as well as the processed data graphed individually and collectively; and finally, a power point presentation with a proper title, all the results and graphs, which is ready to present to any meeting. All this happens in the span of a minute, while the researcher is changing to the next sample. Needless to say, all this data is saved within recursive folders, each properly named according to the different parameters requested to the user by the application.

So, when combining the two solutions, we decrease the measuring time from what used to be, for a really small 4 device batch, around 2 hours of measurement and an additional 4-5 h of data processing, into 24 min of measurement and 4 minutes of data processing. We can easily see that this setup does not only accelerate the pace for any given measurement but it allows us to be much more ambitious with our experiments without fearing having to spend countless hours measuring all our devices. Furthermore, it reduces the amount of time the solar simulator lamp needs to be on, reducing both energy consumption, and degradation; it ensures the reliability of the cell connections thanks to its gold coated pins which perform much better than traditionally used crocodile connectors; and it allows the sample to be measured easily on inaccessible locations, thanks to the remote actuation.

3.1.2 Hardware: Circuit Design and 3D prints

The physical setup consists mainly on a bistable relay based demultiplexer a controller and a 3D printed holder with gold plated pogo pins that contains the solar cell. The circuit itself has undergone several upgrades throughout this work, that were focused on convenience, operation reliability and measurement accuracy. The circuit is based on a binary relay tree, which consists on a sequence of relays sequentially arranged so that each layer of the tree only needs one control signal. In this way a small cheap controller, such as an Arduino, can control dozens of relays with a small amount of logical outputs. In our circuit we are using a total of 23 relays controlled by 10 outputs, which are actually 5+5 due to the fact that our relays are bistable. The main advantage of a bistable relay is that, as its name implies, it has two stable states. That means that when the relay receives power to switch, once it switches, it stays in that position, even after the power is removed, similarly to the usual light switches we find on the walls This configuration poses a series of advantages such as the fact that we can switch all the relays layer by layer, instead of all at once, greatly reducing the maximum power needed to switch them. This is important, because in the first circuit versions, the relay power was directly supplied by the Arduino itself, which has a very limited maximum power output of around 1 W, which was incapable of switching all the relays at once. This has been somewhat mitigated in the latest version, by providing an external power supply option and a transistor to provide power to each relay layer, however the option to power the board via the Arduino is still there so we want to keep the possibility of sequential switching to prevent any power shortage issues. Another important advantage of this configuration is that, when a bistable relay switches and changes to its new state, we do not need to keep on providing power to keep it in that new state, which might seem like a small power saving for only one relay, but taking into account that we have 12 relay pairs and each one draws 200 mW, turning them on for only a few seconds during the entire measurement routine dramatically reduces the total power consumption of the board.

We chose to use mechanical relays instead of any of the available solid-state alternatives due to the fact that mechanical relays provide an ohmic electrical connection between the sample and the Sourcemeter. That allows us to measure a great variety of devices both in alternating current (AC) and direct current (DC) regimes, with a wide current range from subnanoamp to the maximum rated current of the relays of 1 A. It also provides a wide frequency range, where the frequency is not limited by the relays but by the board itself, allowing for high frequency measurements without any signal distortion. In fact, these relay’s main application is in RF communications due to their superior performance at those ranges.

The substrate that we use has a pixelated electrode array that provides a total of 24 pixels which can be measured, 12 on each side. However, each pixel has its negative and positive electrode. The positive electrode sticks out the side but the negative electrode runs down the middle of the substrate. That means that we only need one contact for the negative, or common ground signal, and we need one contact for each of the positive electrodes, or signals. With that approach in mind we designed our first substrate holder with 28 pogo pins which contacted the 24 pixels + 4 connections on the sides for the common signal ground. Nevertheless, thanks to the studies of Dr. Enrique Pascual we found out that since our negative electrode is made of ITO, which has a resistance of 20 Ω · □-1, it was introducing a significant amount of series resistance to our devices, hindering their performance. That is why we decided to take advantage of the small common ground signal electrodes that run in between each signal electrode and we added extra pogo pins to measure the negative electrode as close to the sample as possible. In order to do so we also opted to use commercially lined pogo pins, that come already embedded in their plastic holder and have the standard pin spacing like the one of our substrates. This also made it easier to fabricate the 3D printed holder, which at first it had to contain each pogo pin separately, but now it just has to accommodate the entire pin row, making the manufacturing tolerances much looser.

The 3D printed holder is carefully designed so that we can measure standard 75 mm x 25 mm pixelated substrates either if they are encapsulated or not. To achieve that we have added some relief spaces on the front and the back to accommodate for any inconsistencies during the encapsulation process. We have also added some small ramps so that, if a substrate were to get stuck, we could access its rear side and push it out with some leverage. The holding mechanism is based on four cantilevered snap fits that when pressed against each other hold the substrate in place. The lid itself has the four cantilevers which, being 3D printed have a small degree of flexibility that allows them to snap on to the mating part when pressed. These same cantilevers have a leaver behind so that they can also be easily released to remove the substrate from the holder. The lid also has a 3D printed gasket which is highly flexible and prevents substrate damage when under pressure. Finally, the pogo pins themselves impart a lot of pressure on the cell guaranteeing a good electrical contact.

The pogo pins are soldered directly onto the PCB but the substrate holder, being 3D printed needed to be attached in some other way. After careful consideration and several trials with glues, we devised a method that combined the same attachment strength as a glue, with the same repairability and reversibility as a solder joint. This attachment method consisted on melting a bare header pin into the plastic with the soldering iron that, once cold stayed firmly attached. Afterwards this header pin was soldered onto the PCB as if it was any other electronic component. In this way, if we needed to chance the cell holder for any reason, we just had to desolder 4 header pins and place the new holder in place. An added advantage of this method is that, since we did not need to attach the pogo pins to the 3D printed holder, because they are both firmly attached to the same PCB, if there is the need to change the 3D printed holder we do not need to desolder all the pogo pins, making the board easily repairable in case of damage.

The board itself has had 3 major versions. The first one, which was fully manufactured within our facilities, was the first functional prototype. The manufacturing process consists on exposing a copper bare PCB with a photosensitive resin to UV light, in our case we used our solar simulator, through a shadow mask with the desired circuit layer. Afterwards, we need to remove the exposed resin (in the case of a positive resist) with a solvent, and then etch the copper with an acid to remove the exposed regions and end up with the desired circuit layout. This process was a bit crude and some small corrections had to be made afterwards with a bit of solder. Finally, we had to drill all the holes for the components and solder them in place. Since this process is time consuming and really prone to manufacturing errors we decided to manufacture our second and third versions with a professional manufacturer (JLCPCB).

The main changes between version one and version two are the full board integration of the controller, where the Arduino was integrated within the board itself. And the relocation of the substrate holder on the centre of the board, as well as the addition of the pogo pin connections in between the signal electrodes for the common signal ground connections, and a couple connectors for temperature measurements of the substrate. The main changes between version two and version three are; The addition of controlling transistors, that handle the switching of the relays relieving the controller from having to handle the power-hungry relays and freeing some control pins for other purposes; The addition of a common signal ground line completely separate from the board ground, to minimize noise for high accuracy measurements; The addition of two BNC connectors, one for signal and one for ground signal, to further minimize noise; The addition of switch buttons on board so that the board can also be manually operated; The possibility to power the relays with either the Arduino or a completely separate power supply to further relieve the controller from power delivery; The addition of self-test functionality to check the connection of each cell separately in a fast and reliable manner; The addition of drilled holes for mounting screws; The addition of external header pins for the possibility of outside control from any other controller, either 3.3 V logic or 5 V logic. The addition of substrate labels to clarify correct substrate orientation; The addition of aesthetic labels with the logo of our group; And the addition of the webpage where this board can be found as well as a QR code that brings you directly to that page when scanned.

The control system to select the cell is relatively simple and it is based on only 5 bits. With these bits we can see that, in theory, we could produce up to 32 different states, but in our case some of those states are not used due to the fact that our relay tree is unbalanced. That is why in the end we only use 24 of those states. To select a specific pixel, we need to work out the state of each relay layer so that the desired cell ends up connected to the signal output pin. For example, if we want to select pixel number R.03 we will need to set our relay layers to, from top to bottom, 00101 . And we can also see that, by changing the last bit, to 00100, we now have selected cell R.04. With this method we can easily select each and every cell with just 5 control signals. We want to remark that, because we use bistable relays, writing a 0 on one relay line does not bring it back to the original state. That is why in our specific case, we needed to dedicate a second, negated control signal with also 5 bits which are the complementary bits of the control signal. In this way, when writing a 0 to the Set control signal, we will write a 1 to the Reset control signal, returning that relay to the original position. An easier approach might have been to reset all the relays with just one reset control signal every time, but given that we still had plenty of outputs in our Arduino, and that would have led to higher power consumption and unnecessary mechanical cycles for the relays, we chose to use the more control pin expensive approach.

3.3 Software and operation

The demultiplexing board is controlled with an Arduino Nano, a very inexpensive and accessible microcontroller that can be bought for ~20€ on the official site or for ~2€ in the case of clones, which provide exactly the same functionality. We have chosen to use such microcontroller because it is completely open source both on the hardware and on the software end, allowing for an extremely high degree of tunability and modifications. This microcontroller runs scripts that are usually developed using a small variant of C++ code, which can be written and compiled for the Arduino either within their own, completely free IDE, or other coding software such as Visual Studio. The firmware to control the board was fully developed within this work using the Arduino IDE and it is completely open source.

The IV Characterization software, on the other hand, was developed using the graphical language LabView. This language, although it is not open source and requires a license for developing and distributing scripts and applications, excels in the development of Graphical User Interfaces (GUI) both in simplicity of implementation and functionality. That is why we chose to sacrifice the accessibility of this part of the project in favour of a higher quality, and more intuitive resulting software. Besides we also use this software to communicate with other licenced software such as Power Point, Excel and OriginLab, so it did not make a lot of sense to go through al lot of effort trying to keep this small part licence free while the external software still requires licences.

3.3.1 Pika Demultiplexer Firmware

The firmware for the Arduino that controls the demultiplexer board is a relatively simple piece of firmware that consists in approximately 500 lines of code, including spaces. The code is extensively documented so that, anyone, even with little to no knowledge of C++ or any programming language can, more or less understand what each part of the software is doing.

The code can be broken in 3 main parts: The start, the Get Command and the Command Analyzer. The start code is just a formality which initiates the serial communications between the Arduino and whatever it is connecting to so that the Arduino can correctly receive commands. In this code the Arduino also sends a handshake message where it identifies itself by sending the following string of characters: *“Arduino Multiplexer Pika”*. In this way we can tell that Arduino apart from any other instrument that uses similar serial communication ports. Finally, it also correctly sets the digital pins of the Arduino as outputs, to control the relays, or as inputs, to measure any connected sensors, depending on what the user specified on the IDE.

After a successful start and correct identification, the Arduino begins looping between the Get Command and the Command Analyzer phases. In these phases the Arduino is listening on the serial port for any incoming signal. When we send any character through the serial port, the Arduino immediately reads it and stores it within a string of characters. When we send an “end line” or “carriage return” character, the equivalent of pressing the enter key on the keyboard, the Arduino understands that we have finished sending our message and proceeds to analyse it.

In the analysis phase there can be 4 possible outcomes. The first and most common will be that, when we send a command with the desired pixel label, for example “*R.03*” the Arduino will extract the information from that string of characters, read its own internal database to see which combination of relays connects that specific pixel to the signal output, and actuate each relay line accordingly. All that will happen in under 2 seconds. The second most common outcome is that we send a misspelled command, or a pixel label that does not exist, such as “*R.0.2*” or “*R.14*”, in that case the Arduino will respond us via the serial communication port with an error message: “echo: Unknown command: *R.0.2”* telling us that that command is not recognized. To mitigate frustration for the user we added a third possible outcome that sends a “help” string through the serial whenever the user types in Help, help or simply h. This string of characters reads a small help guide meant to clarify the usage of the software for any confused user.

*“Label commands: Left or Right from 01 to 12 (R.01 , L.07 , R.12)"*

*“Test command: Performs a sweep over the whole range of connections (Test, test)"*

The final outcome of the analysis phase is most useful for debugging purposes, to check if every relay is in good shape and all the pixels can be connected properly. We refer to it as the Test Run and it is basically a small subroutine that connects every pixel to the output signal sequentially for a specified number of seconds so that the user can manually check the connection. To accelerate the connection check, we have manufactured a test PCB that emulates one of our substrates, with the same shape, size and connector layout, but the main difference is that it has small LEDs soldered to it. In this way, when we activate the small debugging switch that we have on the board, we connect the signal output to 5 V and the signal ground to ground, essentially supplying 5V to whichever pixel is connected through the signal line. When we run the test we easily see that each led lights up once for every switch, which indicates that the self-check was successful. In case some of the LEDs did not light up or light up more than once we see there is a problem with the board and it needs to be repaired. The small self-check switch is covered with a small plastic lid to prevent anyone from switching it by mistake.

3.3.2 IV Characterization Software

This LabView application is a piece of software that has been continuously developed and upgraded over the span of this work. When we execute it, the first thing we see is an initialization prompt that informs the user about the current initialization step the software is in. This initialization process mainly resets all variables to their starting state, reads configuration files and automatically detects all the equipment needed to perform the measurements. It also makes sure that the external programs needed for data processing and report generation, i.e. OriginLab, Excel and Power Point are correctly installed. In case there is some error, such as some missing equipment or some of the software not installed, the user receives a prompt to connect the missing equipment or to install the missing software before continuing. This prompt allows the user to either retry, in case of reconnecting an equipment, close the software, to install the necessary data processing software, or to continue with limited functionalities, a status very useful in case we do not need a report or a certain piece of equipment. For example, we could just want to measure one single solar cell, so we would not need the Pika Demultiplexer, or we might not want to generate a report, so we do not need to have OriginLab installed.

After initialization the user is presented with the main Graphical User Interface (GUI) which consists on a variety of controls and tabs. The main tab allows the user to switch between the acquisition panel, the refined results panel and the configuration panel. The acquisition panel is the main panel where the user begins, in this panel we are presented with a graph that will display all our JV curves and a variety of indicators and controls that will help the user make the most of her measurement. These controls allow the user to vary anything from the voltage and current range, to the cell area, the sample name or the measurement date. The latter two belong to the sample naming section, which is of particular importance to ensure the correct management of the high volumes of data generated by this piece of software. Before being able to acquire any data, the user is forced to introduce their name acronym and a batch number to make sure this measurement is stored within the proper directories automatically.

The user is also usually required to select which pixels he wants to measure. For that there is a control to the right of the screen which shows a schematic view of the substrate where the user can individually select each pixel separately, only one row or all pixels. It also gives the user the possibility to select the option No Multiplexer, for cases where the multiplexer is not needed and to randomize the measuring order of the cells to prevent any sequential degradation from appearing as a pattern within the measurement.

Another optional feature the software provides is to add the layer information for each of the deposited layers. This information is easily added through a pop-up window that allows the user to select the type of layer (HTL, AL, BE …) and all the information related to it. In this way the final report also contains this information which has proven to be really useful to compare not only between individual samples but between different batches and even between different users. This can be done thanks to the Nanopto Database developed within our group by Dr. Enrique Pascual, which stores all the necessary information for each batch in an enormous database that can be used to compare between all devices manufactured within our group. Furthermore, with this data my fellow colleagues Dr. Xabier Rodriguez and Albert Harillo have been able to train deep learning models to predict the relationship between different parameters of our devices.

After adding all the desired information to the software we run the JV Curve acquisition in one of 5 modes. Modes 1-3 (Rough, Normal, Fine) are basically varying degrees of accuracy both in integration time per point and number of voltage points. Mode 4 (Custom) is the mode that allows the user to tweak all the parameters exactly as he wants. And finally, Mode 5 (Smart) is an experimental mode, which does not perform a normal JV Curve but rather searches for the “interesting segments” of the JV Curve and scans with high precision only on those regions. These interesting segments are the region where the curve crosses both the X and the Y axis, where we find Voc , Jsc , Rsh and Rs , and the region where the power delivered by the device is maximum, or Vmpp and Jmpp. To search for these points it basically forces the V=0 condition and measures the current and does the same for the J=0 condition. Once it knows those points it performs a fine scan over those regions to more accurately determine the crossing point and the slope within that region. And finally, knowing that the Vmpp and Jmpp must be somewhere between those points it uses a series of halving approximations to find points closer and closer until it finds the best candidate and performs a small scan to guarantee that is the local minimum.

Regardless of the mode used, the output is a JV Curve, which is immediately processed to extract the most relevant parameters (Efficiency, Filling Factor…). After all the measurements have finished the user is prompted to save or discard the measurement. It also has a third option to take a deeper look at the data, which delays the prompt for 30 seconds so that the user has more time to look more deeply into the data. In case the user decides the data is worth keeping, it is prompted again to select if it wants a full report or just the .txt files. A full report will take approximately 1 minute to generate because of the back and forth interaction between LabView and the external needed software. This full report will consist on an OriginLab file containing all JV Curves within a folder in different Worksheets with their appropriate short and long names, their corresponding graphs, as well as all the most relevant parameters on a different folder with their corresponding graphs plotted for each individual pixel. The plots also include a brief description of the layer information within the plot so that the user can easily identify which device performance is related with each layer parameter. The report also includes an excel sheet with the most relevant parameters as well as all the layer information so that it can be easily imported into the database. And finally, it includes a Power Point presentation that summarizes all the information about the measurement results, ready for presentation. After the report is generated and all the files are saved within their respective folders the user is prompted one last time, informing her that she can proceed with the next measurement or close the program.

3.2 External Quantum Efficiency Characterization Setup