Automated IV-measurements for solar panels in stratospheric conditions

Lowe Lundin
Dag Lindgren
Lowe Erickson
Abstract

Automated IV-measurements for solar panels in stratospheric conditions

Lowe Lundin, Dag Lindgren, Lowe Erickson

In this project an automated IV-measurement circuit able to withstand stratospheric conditions was developed. It was made to be part of an experiment which will be executed at an altitude of approximately 27 km. The IV-measurements aim to conclude the power generated by the solar panels during the flight. The circuit is able to adapt to different solar panels by use of a feedback circuit that is able to adjust the voltage through varying resistor ratios. Different solar panels have been tested with the circuit in sunlight and the results show that it is successful in performing the intended measurements. The circuit has also been tested in -80°C as well as 0.01 kPa with no reduced functionality, which means it is well fitted for stratospheric conditions.
Populärvetenskaplig sammanfattning

Denna rapport behandlar utvecklingen av spännings- och strömmätare i syfte att användas för insamling av måtdata från Uppsalautvecklade CIGS-solpaneler i stratosfären. Sex solpaneler kommer sedan tillsammans med mätutrustningen att skickas med en väderballong till ungefär 27 km höjd för att avgöra solcellernas tålighet mot kosmisk strålning. Den ungefärliga tiden för färden kommer att vara 3-5 timmar.

Contents

1 Introduction 3
  1.1 Context ................................................. 3
  1.2 Problem description ................................... 3
  1.3 IV-measurement ........................................... 4
  1.4 Hardware Theory ....................................... 5
    1.4.1 Arduino MEGA ....................................... 5
    1.4.2 Arduino Nano ........................................ 5
    1.4.3 Temperature module .................................. 5
    1.4.4 Printed circuit board ................................. 6
  1.5 Software .................................................. 6
    1.5.1 LTSpice ............................................... 6
    1.5.2 KiCad ............................................... 6
    1.5.3 Arduino language .................................... 6
    1.5.4 I²C-Protocol ......................................... 6

2 Development 7
  2.1 Planning ................................................ 7
    2.1.1 Solar panel ......................................... 7
    2.1.2 Measurement part .................................... 8
    2.1.3 Voltage amplifier .................................... 8
    2.1.4 Current amplifier .................................... 9
    2.1.5 Signal control ...................................... 9
    2.1.6 Signal feedback ..................................... 9
  2.2 LTSpice .................................................. 10
  2.3 Breadboard testing ...................................... 10
  2.4 KiCad .................................................... 10
  2.5 Production of PCB ....................................... 12
  2.6 Software ................................................ 12
    2.6.1 Arduino Nano code ................................. 12
2.6.2 Arduino MEGA code ........................................... 13

3 Results ........................................................................ 13
  3.1 Tests in sunlight ......................................................... 13
  3.2 Testing the BMP280, temperature/pressure module .......... 14
  3.3 Temperature test ...................................................... 14
  3.4 Pressure test .......................................................... 15

4 Discussion .................................................................... 15
  4.1 Tests in sunlight ......................................................... 15
  4.2 Automation ............................................................. 15
  4.3 Durability in stratospheric conditions ............................ 15

5 Conclusion ................................................................... 16
  5.1 High current low voltage solar panel ............................ 16
  5.2 Measurement accuracy .............................................. 16
  5.3 Altitude adjustments ................................................ 16
  5.4 Software efficiency .................................................. 16

Glossary ......................................................................... 18

A Arduino Nano Code ..................................................... 20

B Arduino MEGA Code .................................................... 23
1 Introduction

1.1 Context

The project the group is undertaking is part of a bigger experiment. In the autumn of 2017 Hugo Laurell assembled a team of four fifth grade students in the masters programme in engineering physics. Together they set up the experiment LODESTAR which aims to measure cosmic radiation induced defects in CIGS solar panels. The team then presented the experiment to the REXUS/BEXUS-programme jury in the Netherlands and got accepted into the programme. The BEXUS launch, on which LODESTAR will be featured, is a Balloon Experiment for University Students. The balloon will take the experiment, along with three other experiments on a "gondola", up to an altitude of approximately 27 km above sea level.

After having been accepted into the launch, the team had to increase the number of participants in order to make the experiment a reality before the launch in October of 2018. Nine of the new members are third grade candidate students on the engineering physics programme, which were divided into groups:

- Radiation group, responsible for building a board for a Geiger counter that measures the incoming cosmic radiation.
- Mechanics group, responsible for building the box/outer shell which will contain the experiment.
- Circuit design group, responsible for collecting data of the effect produced by the solar panels during the flight as well as gathering data from the Geiger counter and a thermometer/barometer.

The group writing this report is the circuit design group. Three of the new members are fifth grade students, who are responsible for simulations in the COMSOL simulation software, communication software and solar cell theory including DLTS-measurements (for detecting electrical defects in the CIGS solar panels).

1.2 Problem description

The aim of this project is to create an automated circuit measuring voltage and current over varying loads connected to six individual solar panels. The measurements will be logged by two Arduino Nanos which communicate the data to an Arduino MEGA. The circuit must also be able to withstand
stratospheric circumstances which means that it must not only work in temperatures as low as $-80^\circ$ and pressure as low as 2 kPa but also have an effect generation limited to 400 mW, as heat transfer by convection will be made impossible by the low pressure. The circuit must also be able to scale the measured signals analogically so that the largest possible measurement is just below 5V since the sensors range is 0-5V. This will improve the precision since the Arduinos have 10 bits analog to digital converters (ADC’s), meaning that each step is $5V/2^{10} = 4.88mV$. Furthermore the circuit must be flexible so that it can easily be adapted to different solar panels.

1.3 IV-measurement

![Flowchart](image)

Figure 1: Flowchart showing data processing. The green parts and the measurement of the radiation data will not be covered in this report.

The current/voltage (IV)-measurements are performed on the solar panels to monitor the power generated by the panels. The 6 panels are placed in groups of 3 upon 2 circuit boards, which both contain an Arduino NANO. The NANO’s will gather the data through their analog ports and then transfer the data using the inter-integrated circuit (I$^2$C)-protocol to the
Arduino MEGA. For the project, the data will then be read by a computer connected to the MEGA by use of an USB-cable. This, however, will not be the case during flight.

1.4 Hardware Theory

1.4.1 Arduino MEGA

The Arduino MEGA 2560 is a microcontroller board. The microcontroller is easily programmable by a software called Arduino developed by "The Arduino project", to upload the program or "sketch" as Arduino calls it, it is possible to set up a USB connection between the microcontroller and the PC from which the software is utilized. Besides a USB connection the following ports are used; 5V to power the Arduino nanos, ground, SDA and SCL pins for communication between the different devices. The Arduino MEGA 2560 was chosen to be the projects core unit as it has the capacity to store larger amounts of data if needed, as well as making processing faster. This will optimally ensure that the transitions between gathering the data from the measurement units and sending the data back down to earth can be done without any interruptions.

1.4.2 Arduino Nano

The Arduino NANO 328 is a microcontroller board. The Arduino Nano 328 shares a large portion of features with the Arduino MEGA board but is smaller in size as well as in memory and processing power. Used in the experiment are five volts pin, ground, three digital pins for PWM and eight analog pins of which six are for measurements and two for SDA and SCL communication. The Arduino Nano 328 was, for the measurements, chosen over the MEGA as it fulfills the requirements while also being cheaper and having a smaller size.

1.4.3 Temperature module

The temperature is measured using an Adafruit BMP280, that is able to measure both temperature and pressure. This project only uses the temperature function of the chip, which is able to measure between -40° and +85°. The chip communicates to other devices using the already mentioned I²C technology to transfer measured data. The following pins will be used; SDA, SCL, ground, Vin.
1.4.4 Printed circuit board

A Printed Circuit Board (PCB) connects electrical components using tracks etched onto the board. Almost all components on the PCBs in this project will be Surface mounted devices (SMDs), components which are soldered directly onto one side of the PCB and then connected by the tracks on the same side. The tracks can also run through bridges, which are wires spanning the backside of the PCB. The only non-SMD components are the MOSFETs which are through-hole.

1.5 Software

1.5.1 LTSpice

LTSpice is a computer program designed to enable schematic drawing and simulation of electronic circuits. The potential of certain nodes can easily be deduced, but more advanced features such as voltage sweeps are also available. The program makes it easy to try out circuits and edit values of components in order to optimize functionality.

1.5.2 KiCad

KiCad is a program made to aid designing Printed Circuit Boards (PCB). By drawing up the electronic schematic and assigning the components "footprints", which describe their measurements as the circuit is realized, the program can allow a click and grab interface with which the PCB can be formed. When the components have been placed the program also contains a feature for drawing the tracks which connect the components.

1.5.3 Arduino language

The Arduino language is built on top of C++ and comes with prebuilt functions for C++. The software Arduino is just a text editor that runs all code through a C++ compilator with a library of the pre-built functions[2].

1.5.4 I²C-Protocol

I²C or Inter-Integrated Circuit is a protocol used to transmit data between computers. It allows communication between several slave devices and one or more master devices. In this project it is used for communication between Arduinos’ by importing the wire library and utilizing the SCL (clock line) and SDA (data line) port.
2 Development

2.1 Planning

Figure 2: The configuration of the circuit as it looks in the final LTSpice schematic

Each IV-measurement part of the circuit can be divided into six parts. These parts are explained below and the component names are shown in figure 2. Inspiration for the circuit was fetched from Zimmermann & Edoff (2012) [1].

2.1.1 Solar panel

In the virtual circuit shown in figure 2 the solar panel had to be simulated. In figure 2 the voltage source, called "IL", and the diode to its right simulate the voltage generated by our solar cell. $R_{\text{shuntsolar}}$ and $R_{\text{seriesolar}}$ in figure 2 are intrinsic parasitic resistances that exists in all solar cells and are implemented in the LTSpice simulation as resistors to simulate the behavior of a real solar panel. In an ideal solar cell the $R_{\text{shuntsolar}} = \infty$ and $R_{\text{seriesolar}} = 0$, but as that is not the case they are given realistic values that correspond to the tested solar panels. The E1 and F1 components in figure 2 simulate the amount of layers and area of the solar cell respectively.
2.1.2 Measurement part

The measurement part consists of a MOSFET(2N7000) connected in series to a resistor called $R_{\text{shunt}}$ with a grounded node in between. The voltage of the solar cell is measured over these two components while the current is measured by taking the potential between ground and the Solar- node and dividing by $R_{\text{shunt}}$’s resistance. $R_{\text{shunt}}$ needs to have a very small resistance to allow a good measurement of the approximate short-circuit voltage/current. The MOSFET allows varying the load on the solar panel by varying its gate voltage. This allows the load to vary from $R_{\text{shunt}}$’s resistance + the intrinsic resistance of the MOSFET to infinity. To prevent a worst case scenario of the current flowing in the opposite direction a MOSFET with a diode is used. A capacitor between the solar+ node and ground is needed to prevent oscillations in the circuit.

2.1.3 Voltage amplifier

The voltage over the solar cell is measured by amplifying the voltage over the solar cell with a differential amplifier. This increases the accuracy of the readings as the Arduino is not suited for the low voltages over the solar panels. The voltage amplification is calculated below.

\[
\begin{align*}
I_{R5} &= \frac{V_{\text{solar}} - V_{a1}}{R_5} \\
I_{R1} &= \frac{V_{\text{solar}} + V_{b1}}{R_1} \\
I_{R3} &= \frac{V_{a1} - V_{\text{scaledV}}}{R_3} \\
V_{a1} &= V_{b1} 
\end{align*}
\]

(1)

\[
V_{b1} = V_{\text{solar}} + \frac{R_2}{R_1 + R_2} 
\]

(2)

\[
V_{\text{scaledV}} = -V_{\text{solar}} \frac{R_3}{R_5} + V_{\text{solar}} \left( \frac{R_2}{R_1 + R_2} \right) \left( \frac{R_5 + R_3}{R_5} \right) 
\]

(3)

\[
\begin{align*}
R_5 &= R_1 \\
R_2 &= R_3 
\end{align*}
\]

(4)

\[
V_{\text{scaledV}} = \frac{R_3}{R_5} (V_{\text{solar+}} - V_{\text{solar-}}) 
\]

(5)

The general equations for the differential amplifier are described in equation 1-3. If resistors are chosen so that equation 4 is true, equation 5 follow as a consequence from equation 3. The resistors can thereafter be adjusted, as is needed to gain a voltage suitable to read with the Arduino.
2.1.4 Current amplifier

The current amplification is calculated below.

\[
\begin{align*}
I_{R_8} &= \frac{V_{\text{solar}} - V_{a2}}{R_8} \\
I_{R_7} &= \frac{V_{a2} - V_{\text{scaledC}}}{R_7} \\
V_{b2} &= 0 \\
V_{b2} &= V_{a2}
\end{align*}
\]

(Eq. 6)

\[
V_{\text{scaledC}} = -V_{\text{solar}} \frac{R_7}{R_8}
\]

(Eq. 7)

The current of the solar cell is measured after amplifying the voltage over the static resistance as to increase accuracy, since the Arduino can’t make a reliable reading given the small currents running through the circuit without amplification. To amplify the voltage, an operational amplifier with an amplification of \(R_7 / R_8\) is used. This is then measured as a voltage in the Arduino, given this a current is calculated by dividing the amplification factor and static resistance and \(R_{\text{shunt}}\). Equations 6 and 7 can be derived as the operational amplifier strives to reach an equilibrium between its minus and plus inputs. Depending on the difference of the inputs the output voltage is increased or decreased.

2.1.5 Signal control

To control the load over the solar cell the PWM output of the Arduino Nano is used. The PWM port is capable of sending a square wave with 256 different square lengths and an amplitude of 5V. This square wave is scaled by a voltage divider and then filtered through three low pass filters giving a very steep slope of the transition band (-60 dB per decade). This will filter out all other frequencies than the fundamental frequency which will be the same as the signal’s mean value over a period. In this way a steady static voltage is achieved in the negative input of the operational amplifier.

2.1.6 Signal feedback

The static voltage after the low pass filter, is the desired voltage over the solar panel. To achieve the desired voltage the output of the operational amplifier is connected to the gate of the MOSFET-transistor, which is acting as a variable resistor in the circuit. The positive voltage of the solar cell is connected to the positive input of the operational amplifier. The desired voltage put into the negative input of the operational amplifier and
the output of operational amplifier combined lowers the output voltage of
the circuit if the desired voltage is higher than the current voltage, effectively
increasing the the resistance of the MOSFET as the gate voltage decreases.
As the resistance is increased, the voltage over the solar cell increases until
the difference of the two inputs of the operational amplifier reaches 0. In
much the same way the voltage output of the operational amplifier increases
if the desired voltage is lower than the current voltage, lowering the resistance
over the MOSFET transistor and then decreasing the voltage over the solar
cell. With this feedback circuit the accuracy of the desired voltage is the
same as the accuracy of the PWM module.

2.2 LTSpice
To check that the theoretical circuit worked, an LTSpice simulation was
created. From the simulations it became evident that depending on the
properties of the solar cell and the MOSFET it would be beneficial to introduce
two voltage dividers. The purpose of the voltage dividers is to see to that
the highest signal voltage is slightly larger than the solar cells closed circuit
voltage. If the highest signal voltage is smaller, a complete IV-measurement
will not be achieved, while if it exceeds it by a large margin, potential measuring
points will be wasted and the increment in gate voltage will become unnecessarily
large. Therefore these resistors should be carefully chosen. Due to the
circuit containing capacitors it takes some time for it to enter a steady
state after a change in the signal voltage. It was determined a settling time
of 25ms was needed to ensure that the circuit had entered a steady state.

2.3 Breadboard testing
After the simulation a prototype was constructed on a breadboard to troubleshoot
the setup. Here two important notes were made. First that the $R_{\text{shunt}}$
resistor must be really small to be able to approximate a short circuit. It
was determined that a 0.1Ω resistor was to be used instead of the previous
50Ω. Second, that to reduce noise, mean value measurements should be
used, this will be further explained in the software part. The tests themselves
looked good and when plotted in MATLAB clearly exhibited the characteristics
of a solar cell.

2.4 KiCad
When the breadboard testing could conclude that the LTSpice-circuit was
correctly designed, the circuit was once again designed in KiCad, though
this time with 3 separate but identical IV-measurement circuits connected to an Arduino Nano. Every component was assigned a "footprint" which describes the measurements of the components when the circuit is realized. The KiCad sketch was then used to design the circuit as it would look on a one sided printed circuit board. It was necessary to utilize bridges, that connect tracks by drilling holes through the board and letting a wire span the backside to connect all nodes.

![Figure 3: 1 IV-measurement circuit connected to an Arduino Nano as drawn in KiCad](image)

The components were then placed in a structure to be able to be realized on a printed circuit board.
2.5 Production of PCB

When the KiCad sketch had been finalized it was turned into "postscripts" which are "shadow-traces" which indicate which surfaces will be covered by copper and which will not. The postscripts were then printed, applied to copper surfaces and then put into an acid in order to etch away the copper surfaces not covered by the postscript ink. The SMD's, the MOSFET’s and the Nano were then soldered onto the PCB.

2.6 Software

2.6.1 Arduino Nano code

The Arduino Nanos has setup code which connects to the I²C network with slave addresses. They then enter a loop and take turns sending data and making measurements. When one of them has transmitted its newest measurement series it will make a new measurement on its next solar panel and then enter a waiting loop, waiting for the Arduino MEGA to make
new data request. The measurements (one current array and one voltage array of size 256) will be sent one int at a time. Since sending an int as a char takes four bytes the program deconstructs the ints into two bytes and then reconstructs them in the Arduino MEGA. This reduces the data on the wire by 50%.

2.6.2 Arduino MEGA code

The Arduino MEGA has a setup that establishes itself as the master in the I²C connection. It then enters a loop where it first asks for data and then receives the measurements from the first Nano and then from the second Nano. It moves on and receives an int from the Arduino Nano connected to the radiation counter and after that it asks the thermometer for a temperature measurement. All communication takes place through the I²C connection and is sent through USB to a computer.

3 Results

3.1 Tests in sunlight

The tests were run with a mix of high voltage, low current and low voltage, high current silicon solar panels, in the midday sun.

![Figure 5: Results for the low voltage, high current voltage, high current left: current plotted over 256 duty cycles, center: voltage plotted over 256 duty cycles, right: voltage plotted over current](image-url)
The low voltage, high current solar panel displayed above had just one solar cell and generated an open circuit voltage of 670 mV and a short-circuit current of around 70 mA.

![Figure 6: Results for the high voltage, low current solar panel](image)

- left: current plotted over 256 duty cycles
- center: voltage plotted over 256 duty cycles
- right: voltage plotted over current

The high voltage, low current panel displayed above consisted of six solar cells connected in parallel with an open circuit voltage of just above 4V and a short-circuit current of around 23 mA.

The results came out as expected with little to no noise. The only thing worth to note is the initial plateau visible in the central graph in figure 5. The cause of this will be discussed later and possible solutions will be recommended.

### 3.2 Testing the BMP280, temperature/pressure module

The BMP280 was tested by connecting it to the Arduino MEGA and running through a measurement cycle, ensuring that it could communicate through the I²C-protocol. The temperature reported from the BMP280 was then compared to a thermometer with no differing measurements, though it is worth noting that the thermometer used for reference only displayed 3 significant figures.

### 3.3 Temperature test

To ensure that the circuit survives low temperatures in the stratosphere the circuit was tested in extremely low temperatures. The circuit was cooled with liquid hydrogen to a temperature of -80°C but could operate without problems. The temperature outside the electronics box is highly unlikely
to fall below these levels and furthermore the circuit will be heated by the effect it produces as convection is virtually nonexistent at the pressure that the circuit will operate at.

### 3.4 Pressure test

To simulate the low pressure that will be present on the 27 km altitude, about 2 kPa, the circuit was tested in a pressure chamber. During the test the pressure was dropped to 0.01 kPa, which is well below the aforementioned expected pressure. The circuit made it through the test functioning with no damage detected afterwards.

### 4 Discussion

#### 4.1 Tests in sunlight

As figure 8 shows, the readings from the small solar panel does not go below 350mV, this is because resistance between the solar panels + and - nodes can never truly be zero. The shunt resistor needs at least to have a resistance of about 1 Ohm to allow any accurate current measurement and the transistor will also have an intrinsic resistance. These resistances multiplies with the short-circuit current gives the lowest possible voltage drop that we can measure.

#### 4.2 Automation

From the tests it can be concluded that the automation of the circuit works as intended. After the Nano’s have gone through the sweeps from 0 to 5V and measured the resulting currents and voltages in the circuits, they are able to transfer the data to the MEGA by use of I²C. This data can then be sent to an external computer in an intended way.

#### 4.3 Durability in stratospheric conditions

The tests where stratospheric conditions were simulated could conclude that the circuits were durable to the strains imposed on them. Although no real sweeps and IV-measurements could be performed due to the lack of sunlight inside the testing facilities, the Arduino’s still communicated with the main computer just as during normal circumstances. After the tests had been performed no defects could be observed, which signals that the circuits will be in no trouble during the flight.
5 Conclusion

The aim of the project was to create a circuit for automated IV-measurements for solar panels in stratospheric conditions. What can be concluded from the results is that the experiment has been successful overall. During the development and the testing several lessons was learned, these are listed below.

5.1 High current low voltage solar panel

The circuit is flexible as it’s possible to change a few resistor ratios to adapt the circuit to a new solar panel. However these changes do not allow for all solar panels. As previously mentioned in the discussion some high current low voltage solar panels will get a large voltage drop (relative to its open circuit voltage) over the shunt resistor and the intrinsic resistance of the MOSFET. Taken to extremes this makes the data collected useless. The plateau can however be lowered by using another MOSFET with a lower intrinsic resistance (the MOSFET in our circuits has an intrinsic resistance of up to 3Ω, there are MOSFETs with intrinsic resistances below 0.01Ω). The shunt resistor might also be lowered somewhat, it should however be noted that this will cause the current measurement to perform worse.

5.2 Measurement accuracy

The Arduinos 10 bit ADC’s gives the circuit 1024 measuring points. However the LM324 op-amps can only amplify the voltage to about 3.56V which leaves 730 measuring points. By changing the op-amps to one which can provide a higher voltage on the output pins (for example TLV27) it would be possible to maintain almost all of the original measuring points.

5.3 Altitude adjustments

The amplification of the current and the voltage is adjusted for sea level altitude. As the experiment is going into space an approximation needs to be done for how the difference in the intensity of sunlight will be. When this is done suitable resistor ratios for the amplification can be chosen.

5.4 Software efficiency

The two parts of the measurement and its processing which undoubtedly takes most time is the circuits settling time when the square wave is changed
and the delay caused by the I\textsuperscript{2}C connection. Therefore two further developments can be identified that might decrease the time a loop takes. Regarding the measurement itself it would be possible to decrease the circuit’s settling time. The current choice of 25ms is quite conservative and it was chosen to make sure that the capacitors would absorb a change from 0V to 5V. However it’s worth to point out two things here. First that the voltage divider might have lowered the maximum signal voltage depending on the solar cells closed circuit voltage, for example in our circuit the signals maximum voltage is 1V. Second that the Arduino Nano only steps up the voltage with 1/256 of the maximum signal voltage per step. This means that the settling time might be considerably decreased, however this should be closely investigated by conducting tests to make sure that the settling time does not become too small. The other development would be to try making the I\textsuperscript{2}C run faster. This would perhaps be possible by overclocking the protocol from 100kHz to 400kHz.
Glossary

ADC  Acronym for Analog to Digital Converter. [18]

BEXUS  Acronym for Balloon EXperiment for University Students, the program on which the experiment is featured. [18]

CIGS  Acronym for Copper, Indium, Gallium and Selenide, the chemical elements featured in the solar panels the experiment will be testing. [11][18]

COMSOL  Program for physics simulations. [18]

DLTS  Acronym for Deep-level transient spectroscopy, a method for studying defects in semiconductors. [18]

Duty  Value between 0 and 255, describes the fraction of a cycle in which a square wave is turned on, 0 means no square wave, 255 means square wave always on. [18]

I²C  Acronym for Inter-Integrated Circuit. [18]

I_{SC}  Acronym for Short Circuit current. [18]

IV  Acronym for Current (usually represented by I) and Voltage (V). [18]

Master/Slave  Communication configuration in which the slave only transfers data after having received a request from the master. [18]

op-amp  Acronym for operational amplifier. [18]

PCB  Acronym for Printed Circuit Board. [18]

PWM  Acronym for Pulse Width Modulation, which is a method for simulating analog signals with digital pulses. [18]

SCL  Acronym for Serial Clock Line, used in the I²C-protocol. [18]

SDA  Acronym for Serial Data Line, used in the I²C-protocol. [18]
SMD  Acronym for Surface Mounted Devices, which are components soldered onto only one side of the PCB. [18]

$V_{OC}$  Acronym for Open Circuit voltage. [18]

References


Appendix A  Arduino Nano Code

```cpp
#include <Wire.h>

int pinV = 0;  //Analog pin measuring voltage
int pinA = 0;  //Analog pin measuring current
int pinGate = 0;  //digital pin controlling the gate voltage

//int period = 256;  //Period time for a square sequence
int duty[256];  //Array filled with values for how long the square part
int trackerLoop = 0;
int trackerReq = -1;  //Keeping track of the current sent data series
int measuredV[256];  //Array storing the measured voltages for a series
int measuredA[256];  //Array storing the measured current for a series
int numberOfMeasurements = 300;
int settlingTime = 25;
int integratingDelay = 30;
byte data[4];
byte mReady = 1;
byte mNotReady = 0;
boolean flag = false;

void setup() {
  Serial.begin(9600);
  Wire.begin(2);
  Wire.onRequest(request);
  //Wire.onReceive(event);

  for (int j = 0; j <= 255; j++) {  // duty cycle of pwm wave.
    duty[j] = j;
  }
}

void loop() {

  switch (trackerLoop % 3 + 1) {

    case 1:
      pinV = A7;  //26
      pinA = A6;  //25
      pinGate = 9;
      break;

```
mReady = 1;
break;

case 2:
    pinV = A1;  //20
    pinA = A0;  //19
    pinGate = 10;
    mReady = 10;
break;

case 3:
    pinV = A3;  //22
    pinA = A2;  //21
    pinGate = 11;
    mReady = 11;
break;

} //void event(int howMany) {

    Serial.print("Measurment series ");
    Serial.print(trackerLoop);
    Serial.println(" initiated ");

for (int j = 0; j <= 255; j++) {

    analogWrite(pinGate, duty[j]); //Creates square waves with the current duty
    delay(settlingTime);  //Circuit is allowed to enter steady state

    int32_t averageV = 0;
    int32_t averageA = 0;

for (int i = 0; i < numberOfMeasurements; i++) {
    averageV = averageV + analogRead(pinV); //Measures V
    averageA = averageA + analogRead(pinA); //Measures A
    delayMicroseconds(integratingDelay); //Prevents interference when measuring
}

measuredV[j] = (int)(averageV / numberOfMeasurements);
measuredA[j] = (int)(averageA / numberOfMeasurements);
if (j == 255) {
    flag = true;
}
while (flag) {
    delay(1000);
}
trackerLoop++;

void intToByte(int a, int b) {
    data[0] = highByte(a);
    data[1] = lowByte(a);
    data[2] = highByte(b);
    data[3] = lowByte(b);
    return;
}

void request() {
    if (flag && trackerReq == -1) {
        trackerReq++;
        Wire.write(mReady);
    } else if (!flag) {
        Wire.write(mNotReady);
    } if (trackerReq >= 0) {
        intToByte(measuredV[trackerReq], measuredA[trackerReq]);
        trackerReq++;
        Wire.write(data, 4);
    } if (trackerReq == 256) {
        trackerReq = -1;
        flag = false;
        return;
    }
# Appendix B  Arduino MEGA Code

```c
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

Adafruit_BMP280 bme;

int measurementV[256];
int measurementC[256];
int temp;
int pressure;
boolean hasMeasurement = false;
int radCount = 0;
boolean radFlag = false;
byte c = 100;
byte aa = 1;
byte ab = 10;
byte ac = 11;

byte measurement_nr = 0;
int endDelay = 0; //delay in ms
int clockDelay = 0; //delay in ms

// Define ground commands
const int GND_CMD_GEIGER_OFF = 0;
const int GND_CMD_GEIGER_ON = 1;

void setup () { 
    Serial.begin(9600); // only for printing
    Wire.begin();
    !bme.begin();
}

void get_radiation_data () { 
    Wire.requestFrom(3, 2); // receives one int from UNO
    byte readR[2];
    int position = 0;

    while (Wire.available()) { // sends an int as two bytes
    ```
readR[position] = Wire.read();
position++;
if (position == 2) {
    radCount = getInt(readR);
    radFlag = true;
}
}
*/

/*
read from nano i
*/

void populate_data_arrays(int i) {
    Serial.print("Reading data from nano ");
    Serial.println(i+1);
    c = 100;
    Wire.requestFrom(i, 1);
    while (Wire.available()) { // slave may send less than requested
        c = Wire.read();
    }
    byte value = 1; // if we read a 1 the series is ready for transmission
    while (c != aa && c != ab && c != ac) {
        delay(2000);
        Wire.requestFrom(i, 1);
        while (Wire.available()) { // slave may send less than requested
            c = Wire.read();
        }
    }
    c=c*2;
    if (i==1) {
        c=c-1;
    }
}
// Read data when nano is ready
for (int j = 0; j < 256; j++) { // Receives a measurementV and a measurementC
    int position = 0;
    byte readV[2];
    byte readC[2];

    Wire.requestFrom(i, 4); // receives one V and one C measurement
    while (Wire.available()) { // sends an int as two bytes
        if (position < 2) {
readV[position] = Wire.read(); //recieves voltage value at j
position++;
else {
    readC[position - 2] = Wire.read(); //recieves current value at
position++;
}
if (position == 4) {
    measurementV[j] = getInt(readV); //assembles the measurments
    measurementC[j] = getInt(readC);
    char* s;
    // sprintf(s, "V=%d, tC=%d", measurementV[j], measurementC[j]);
    Serial.print(s);
    break;
}
}
if (j == 255) {
    hasMeasurement = true;
}

// function converts an array of two bytes into an int
int getInt(byte B[]) {
    int first = (int)(B[0]);
    first = pow(2,8) * first;
    int second = (int)(B[1]);
    //Serial.println("first ");
    //Serial.println(first);
    //Serial.println(second);
    return (first + second);
}

void get_temp_data() {
    temp = bme.readTemperature();
}

void get_pressure_data() {
    pressure = bme.readPressure();
}

int handle_ground_command(char command)
{  
   switch (command) {  
      case GND_CMD_GEIGER_OFF:  
         // turn off geiger tube  
         break;  
      case GND_CMD_GEIGER_ON:  
         // Turn on geiger tube  
         break;  
   }  
}  

void loop () {  
   /*  
      Handle ground commands  
   */  
   
   /*  
      char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  
   */  
   
   int packetSize = Udp.parsePacket ();  
   if (packetSize) {  
      // Read content of UDP package into packetBuffer  
      Udp.read (packetBuffer ,UDP_TX_PACKET_MAX_SIZE);  
      Serial.println ("Contents:");  
      Serial.println (packetBuffer );  
      
      // Define commands  
      
      // Handle ground command  
      char command = packetBuffer[0];  
      handle_ground_command (command);  
   }  
   /*  
      Read measurement data  
   */  
   
   populate_data_arrays (measurement_nr % 2 + 1);  
   // get radiation data  
   // get_radiation_data ();  
   // get temp data  
}
get_temp_data();
// get pressure data
get_pressure_data();

/*@ Send data to ground */
if (hasMeasurement == true) {
    Serial.print("Measurement number");
    Serial.print(measurement_nr / 6);
    Serial.print(" OVer solar panel ");
    Serial.println(c);
    for (int k = 0; k < 256; k++) {
        Serial.print(k);
        Serial.print(" ");
        Serial.print(measurementV[k]);
        Serial.print(" ");
        Serial.println(measurementC[k]);
    }
    hasMeasurement = false;
    Serial.print("The temperature is ");
    Serial.print(temp);
    Serial.println(" Celcius");
    Serial.print("The pressure is ");
    Serial.print(pressure);
    Serial.println("Pa");
}
measurement_nr++;
delay(endDelay);

/*@ Write data to SD-card */
//send radcount
//radFlag=false
//temp/preassure measurement?