**Slide 1: 1st G 2nd M**

Presenting ourselves and the title of the project.

**Slide 2:**

What are the requirements? **G**

* We needed to implement a low-power system that takes environmental data and sends it to a server/mobile app and shows it like graphs.
* This system could be implemented on a moving robot

How many of them did we accomplish? **G**

* Almost all of them.

What is left to be done? **M**

* The only thing that is still left to be done is presenting the data graph-like
* The idea of a moving DAQ is no longer valid

Future work… **M**

* This system can add more sensors, of course
* A CO2 sensor can be very useful if you want to make a intelli-garden
* A system that also takes action with the data collected (ex: waters the garden if the humidity is low, starts the heater if it’s too cold etc.)

**Slide 3: M**

Here we have a brief comparison between three types of devboards.

Frist on the list is the Arduino Uno v3 which is the most popular board of those three. It is used by a lot of people because it is easy to use and quite affordable. The only minuses of this board is that is not a Low-Power oriented board and it does not have a wireless module.

The 2nd board on our list is the RPi3. This is also a very popular dev board but in contrast to Arduino it is not that cheap because it’s a very powerful board. Because of this it uses lots of energy.

The last board on our list is the MSP432. It is produced by Texas Instruments and is low-power oriented. Because this board is low-power it is not that powerful and does not have lots of resources and this makes this board very cheap. The only negative things are that there aren’t a lot of examples online and the board does not have a wi-fi module.

**Slide 4: M**

The cheapness of the MSP432 and the orientation for low-power made us believe that this board suits us the best and we chose to use this TI product for our project.

MSP432 has a lot of low-power modes that we can choose from as listed here but we chose to use LPM3 as our main working mode. We made this decision because LPM3 (also known as ‘Deep-sleep mode’) is a mode that allows us to wake up the device from sleep using a real-time clock and also isn’t a big energy consumer while sleeping.

**Slide 5: G**

The sensors that we used in our project are those two sensors. We chose different types of sensors for educational purposes.

The DHT22 is a Temperature and Humidity Sensor. It is a cheap sensor that is working on a special protocol named 1-wire connection.

The AP3216 is an Ambiental Light Sensor. It is also cheap, low-power and working on I2C interface.

We have also used a temperature sensor that was using ADC, again, for educational purposes.

**Slide 6: G**

For our Wireless communication solution we had to choose between several types. We ended up having to choose between those three:

The Zigbee (XBee): Is a Wireless module that works over Wi-Fi. It has a wide range of action but is not cheap and consumes a lot of energy (140mA).

The fc-114: Is a bluetooth Low-Energy module. It has a medium range of action (up to 30m) and it consumes way less power than the wi-fi, but it is not cheap and is hard to configure (time consuming).

The NodeMCU ESP8266: Is a wi-fi module/devboard. It is a microcontroller that has also a wi-fi module. It is easy to use, cheap and it has a wide range of action (up to 2km). The only downside is that the wi-fi consumes a lot of energy.

**Slide 7: G**

Here we have 2 Charts where we add “numbers” to our comparisons.

The first chart is based on the comparison between our 3 devboards. With the blue color we have the Price for each board (in USD) and with the red color we have the power Consumption (in mA). This graph helped us choose our components because we can conclude that MSP432 is the cheapest and the board that consumes the least power.

The 2nd graph is based on the 3 types of wireless communication modules that we just described. Again, with blue is the Price for each module (in USD) and with red, the power consumption (in mA). This graph didn’t help us a lot while choosing the best component for our project because as we can see the Bluetooth LE would’ve been the best. But for academical purposes, financial and infrastructural resources we have chosen the ESP8266.

**Slide 8: G**

The overview of the system is shown here. We have our main board, MSP432 which communicates on I2C with the ALS, on 1-wire with the DHT and on UART with ESP. The NodeMCU is used as a wireless module to send the data to the telnet Server through Wi-Fi.

**Slide 9: M**

For measuring the energy and power consumption we used a special instrument provided by TI and integrated in the CCS, named EnergyTrace.

Here is an example of the data measured by this CCS feature. We had run a simple LED blink program in LPM3 (deep sleep) and compared the same program running in active mode. We can select the time scale (here we used 10 seconds). You can see that the deep-sleep mode consumes almost 100.000 mJ less than the active mode, resulting on an estimated 4.7 day life of a simple CR2032 battery compared to only 1.7 day life while on active mode.

Beneath we have also the graphs for Power (in mW) and for Energy (in mJ). With green is represented the deep sleep mode and with yellow the active mode.

**Slide 10: M**

Now I’m going to speak about the implementation of this project.

How does the MCU receive and process the data from the sensors?

The DHT22 is a breakout board that can send information about temperature and humidity. The data is sent through a protocol named 1-wire connection. When the MCU is ready to receive data it sends a signal to the sensor. The sensor wakes up and sends back an ACK and then sends 40 bits of data. The first 16 bits is the Humidity and the next 16 bits is the temperature (if the 1st bit of the temp is 1 => the temperature is below 0 degrees Celsius), and the last 8 bits is the checksum. The whole process must beyond 2 seconds. After this, the sensor powers-down till the next signal from the MCU.

The AP3216 is a breakout board which can detect Ambiental Light and also Proximity (if an object is nearby). The sensor uses an I2C Interface to send the data to the MCU. First the MCU sends a Start bit . Then the Slave Address 7 bits + the Write Command. The Sensor sends back an ACK. The next 8 bits are the Register Address. Then we need to send the Slave Address again with the Read command. Then the sensor sends an ACK + the 8 bit AL data. Then the MCU sends an NACK + the STOP bit.

So now, we have the temperature, the humidity and the light data inside the MCU.

**Slide 11: M**

How does the MCU send the data to the NodeMCU?

Because ESP8266 is also a microcontroller we can transfer the data from one microcontroller to another one on the UART protocol. All we need to do is connect one Rx to one Tx and the other Tx to the other Rx. In our case we use the MSP pins 3.2 for Rx and 3.3 for Tx and the NodeMCU software pins D5 for Rx and D6 for Tx.

All we do now is to send the data stored in the MSP432 through the Rx pin to the Tx pin on the NodeMCU. The NodeMCU runs an Arduino Code that reads data on the UART.

So now, we have the data stored in the NodeMCU ESP8266.

**Slide 12: M**

How is the data sent from the ESP8266 to the PC?

On the PC, a C# tcp server is running. The server takes the local address of the PC and the port and waits for connection. The NodeMCU already knows the HostName and the Password and uses them to connect to the tcp Server. Once connected, the NodeMCU can send data through wi-fi to the server.

Now, we have the data on the PC, all that is left to be done is to process the data.

**Slide 13: G**

For those who still have questions here is a short summary of what we have done so far.

* M: Slide 1+2, Slide 3, 4, 9, 10, 11,12
* G: Slide 1+2, Slide 5, 6, 7, 8, 13