Zoltán Kiss - Endrich Bauelemente Vertriebs GmbH

Practical usage of Panasonic GRIDEYE sensor v2.0 – Part 2.

In the previous article of this series the main features and news of the Panasonic GridEye thermopile matrix sensor have been introduced. In this paper we intend to show how to build a simple, low resolution thermal camera using the evaluation board in Arduino and in PC environment. We will create a device that can detect the approaching of a human hand and based on its presence a relay will be operated simulating a gesture-controlled switch. The other sample application will be a PC based control for lighting that can follow the movement of an object.

GRID-EYE sensor as an Arduino shield

A it is known the Panasonic GRID-EYE device is an 8X8 pixel MEMS technology-based thermopile matrix sensor, which is able to detect absolute temperature of 64 standalone section of the space in front of the sensor, based on evaluation of the infrared heat radiation of the objects.

Grid-is able to detect absolute temperature as well as temperature gradient providing low resolution (8X8=64 pixel) thermal image.

To make prototyping easy, Panasonic also developed an evaluation board for the sensor, which is made as an Arduino Due shield.

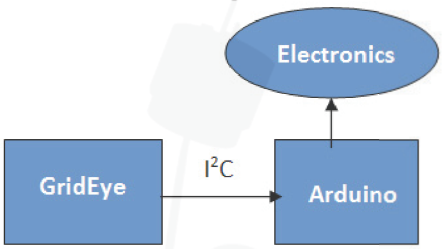
1| Evaluation board made as an Arduino shield

It is important to mention, that the GridEye evaluation board is a 3.3V card, so it can only be used with the Arduino DUE family. To connect to UNO is even not possible as a shield due to the mechanical organization of the board connectors. The Arduino MEGA versions – although they have the same pin layout – are 5V devices, they are not compatible with the GridEye shield due to the different supply voltage levels.

The task

We would like to simulate a simple gesture-controlled switch, that switches on a circuit, when a human hand is present in front of the sensor and switches it off when disappeared. The simulation task requires the read out of the 64 pixels' data through the I²C interface as shown on fig. 2

2| Using I²C bus to supply sensor data to Arduino Due

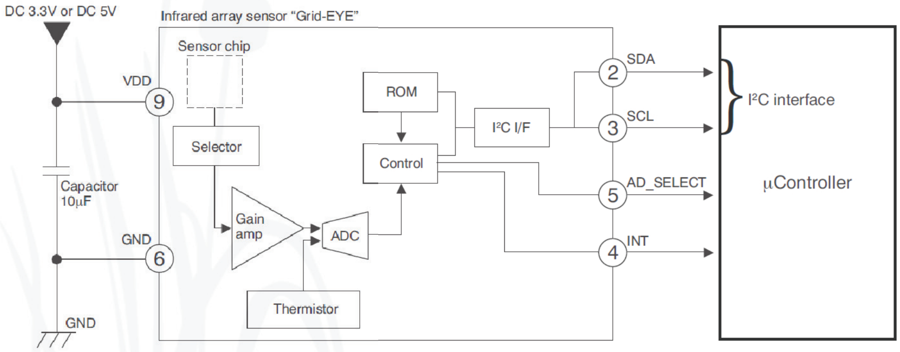


There is also a thermistor available on board, which provides reference temperature of the environment.

To keep the solution simple, we use this value as a reference ambient temperature and compare the average temperature value of all the 64 pixels with it. When a human hand is present in front of the sensor, the pixels' average temperature would exceed at least by 5 °C the ambient temperature measured by the NTC, what we can use to decide whether we should switch the relay on or off.

Arduino DUE

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. This is the first 32-bit ARM core-based ARDUINO family member. It has 54 digital I/O ports, from which 12 can be used as PWM output. We can control a servo motor with it, which can be used to follow the movement of a heat source in front of the sensor. As mentioned the

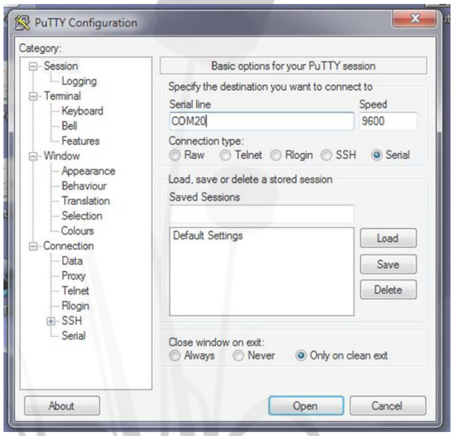


DUE is a 3.3V card, resulting GPIO ports maximum voltage level to be also 3.3V. Due to this fact we often need to match TTL voltage levels up to 5V by using external circuits such as SN74HC125N line drivers (3.3V- 5V).

Panasonic GridEye libraries

The Arduino Library required to support the GridEye Evaluation Kit can be downloaded from Panasonic's support site, and we only need to copy them to the right folder on our PC's Arduino IDE: <Arduino folder>/libraries. Restarting the Arduino development environment, the GridEye libraries are available and we can reference to GridEye.h, GE_SoftUart.h and GE_SoftUartParse.h header files.

GridEye's data structure



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM20

Speed: 9600

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Load, save or delete a stored session

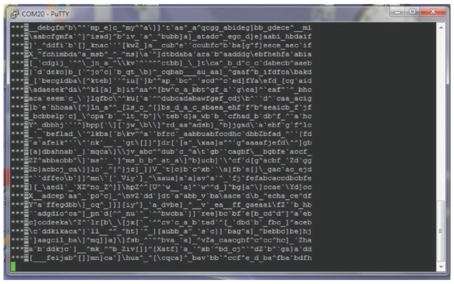
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

3| Configuring PuTTY to connect to development kit via USB

The raw data received 10 times a second on the serial port seems to be chaotic for the first sight, however knowing the structure they can be used easily. Any serial terminal program e.g. PuTTY, running on the developer PC and connected via serial interface to the development kit can display these raw data:



```
***▒__debgfm^b\^^`mp_e]c_^my^^a\]]`t`aa^_a^qcgg_abideg]bb_gdece^__mi
***▒\aabcfgmfa`^]^isad]^b^iv_`a^_^bubb]a]_atadc^_egc_d]e]aabi_hbdaif
***▒]`_^ddfi`b`[]_knac```[kwZ_]a__cub^e``ccubfc^b`ba[g^f]eece_aec`if
***▒X_^fchimbda^a_msb^_^_^ns]\a`^]dtbdaba`ara`b^aadddg\ebfhehfa`abia
***▒[_`cdgij_`^^\_jn_a_^\\kv`^`^^^ctbb]_\_]t\ca^_b_d^c_c`dabecb^aaeb
***▒]`d`deko]b_[`^jo^c]`b_qt_\b]^_cqbab___au_aa]_^gaaf^b_ifdfca\bakd
***▒_[`becgidba\[^kteb]``^iu[`]b^^ap_`bc^_`scd^^c`ed]fYa\efd_[cg`aid
***▒\adaeeek^da\^^kl[a]_b]it^aa^^[bw^c_a_bbt^gf_a``g\ea]^`eaf^`^_bhc
***▒aca`eeem`c_\`]lqfbc\^^ku[`a`^^dubcadabawfgef_cdj\b^``d^`caa_acig
***▒]b`e`hhoaa\[^]ln_a^^_[ls_c_^[]bs_d_a_c_sbaea_ehf`f^b^aeaicb_f`jf
***▒_bcbbelp`c]_\^opa`b`_^lt_^b^]\`teb`d]a_wb`b_`cfhad_b`db^f_^`a`hc
***▒Y^_dbbhj```^]bpp[`\][`jw_\b\\]^rd_aa^adsb]_^b]jgad\`a`ebf`g`f^lc
***▒_`_`beflad_\`^lkba[`b\kv^^a``bfrc`_aabbuabfccdhc`dbbZbfad_^``[fd
***▒`a`afeik^``\^`nk`__`_`gt\[]]^]dr[`[a^_\xaa]a^^`g^aaaafjefd\^^]gb
***▒[a]dbahnab`_]`mqca\]\\jv_abc^^dub`c_^a\t`gb``cagbf\__bgbfe`accf_
***▒ZZ^abbaobb^\]`ms^`_`]^ms_b_b^_at_a\]^b]ucb]`\^cf`d[g^acbf_`Zd`gg
***▒Zb]acbcj_ca\]]lo`_^]^]jz]_]]V_`t]c]b`c^xb``\a]fb`e[]\_gac`ac_ejd
***▒^``dffeo\b`]]^mn\`[`_Viy`]_^\aaua]a`a]av^a`^_`fj`fefabcaccdbcbfe
***▒][_\aedl`_`XZ^no_Z^]]\hpZ^^[U^`w__`a]^`w^^d_]^bg[a^\]ccae`\Yd]cc
***▒X__adcep`aa^_`po^c]_^\nvZ`dd`]dt`a^abb_v`ba\aace`d\b_^echa_ce^df
***▒Y^a`ffegdbb\]_oq^_]]][iy^]_`a_dvbe]_^__v`_ea__ff_gaeaai\fZ``b_hb
***▒``adgdio^ca^]_pn`d[^^_nu`^_`^^bwcba`]]`ree]bc`bf`e[b_cd^d^]^a`eb
***▒c]ccdeeka\^Z^`lr[b\_\[jx[`_`^^cv`c_a_b`tad`^[_`dbd`b`_fbc_]^aceb
***▒\c`ddkikaca^]`ll__^^_^ht]`^_][aubb_a^_`s`c]]`bag^a]_^bebbc]be]hj
***▒`]aagcil_ba\]^mq]]a]\]fsb_^`^^bva_`a]_^vZa_caacgbf^c^cc^hc]_`Zha
***▒a`b`ddkjc`]__^mk_^^b_Ziv[]]^[Xatf]`a_`^xb`^bd_cj^`^dZ`b^`ga]a`dd
***▒[___feijab^[]]mn]ca`]\hua^_^[\cqca]^_bav`bb`^ccf^e_d_ba^fba`bdfh
```

4| Raw data provided by the sensor 10 times a second

135-byte data arrives 10 times a second, some of them are control bytes, that need to be filtered out. The remaining data is the NTC temperature data as well as the 64 pixels' data, which can be fetched out according to below rules:

- 3 bytes header ***
- 2 bytes NTC temperature HL, HH (12-bit signed data)
- 64 X 2 bytes (HL HH) = 128 bytes (12 bit two complemens code)
- CR LF (\r \n)

The task has been simplified now, the data structure is very logical, we just have to take care of the sequence of the bytes, as the low and the high bytes arrive in reverse order.

The conversion required for the calculation of the direct temperature values is as follows:

NTC 12-bit signed value:

- XXXX S _ _ _ _ _ _ _ _ _ _ _

- If S=1, temperature is negative, if S=0 temperature is positive

- The binary value stored on the 12 bits should be converted into decimal and should be multiplied by the resolution of 0.0625oC in order to get the real temperature:
  11 bits = 1024 (211), so over the whole decimal data range (-1023...+1024) it is possible to cover the temperature range of -63 ... + 64 °C.

- Example: HH = 0x01 HL = 2A
  -> Hex-Bin conversion
  ->0000 0001 0010 1010 -> positive temperature, and its value is
  1*2+1*8+1*32+1*256=298 (decimal)
  The temperature is = 298*0.0625°C = 18,6 °C

The pixel values are stored in two's complement code, we need a different way to fetch them out:

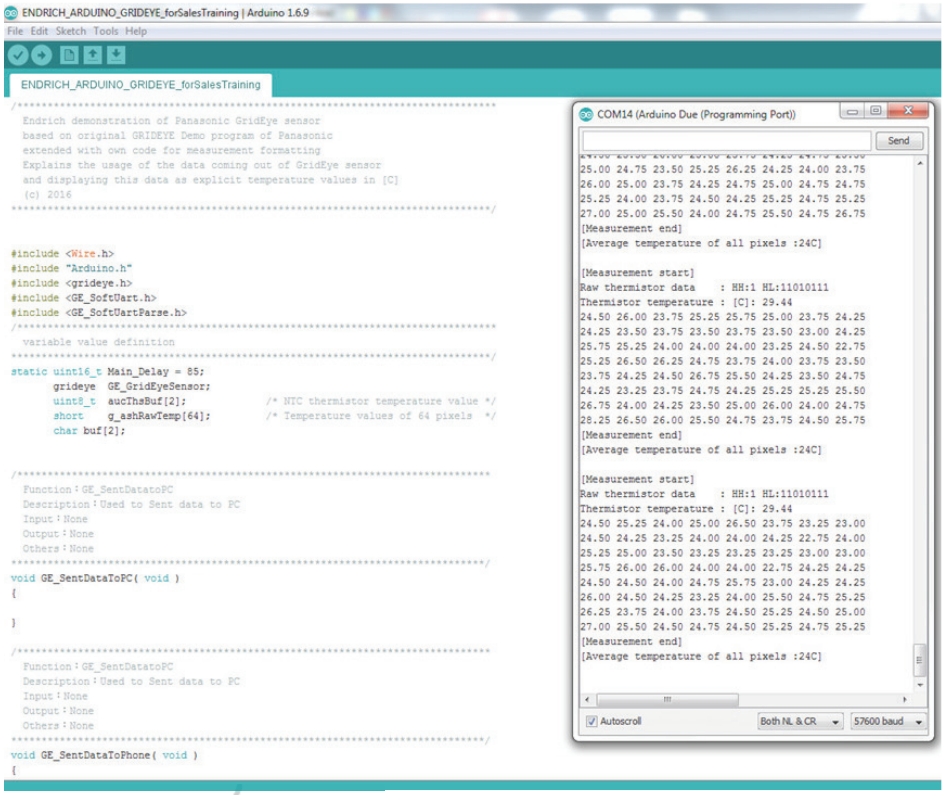
- XXXX S _ _ _ _ _ _ _ _ _ _ _

- The 12th bit refers to the sign as well, however it also holds data: when it is 1, the temperature is negative, so the decimal value becomes -2048: (-212) instead of +2048.

The program reads out the temperature data 10 times a second from the sensor via the I^2C bus and converts the raw data to readable temperatures.

To be able to control the operation of the program we display the results in the built-in serial monitor of the ARDUINO IDE attached to the virtual serial port (in our case COM14).

The Arduino example software written for the conversions can be downloaded here:
http://www.endrich.hu/mintaprogram/ENDRICH_ARDUINO_GRIDEYE_Elektronet.ino


ENDRICH_ARDUINO_GRIDEYE_forSalesTraining | Arduino 1.6.9
File Edit Sketch Tools Help
ENDRICH_ARDUINO_GRIDEYE_forSalesTraining
COM14 (Arduino Due (Programming Port))
Send

```
/*****************************************************************************
  Endrich demonstration of Panasonic GridEye sensor
  based on original GRIDEYE Demo program of Panasonic
  extended with own code for measurement formatting
  Explains the usage of the data coming out of GridEye sensor
  and displaying this data as explicit temperature values in [C]
  (c) 2016
*****************************************************************************/


#include <Wire.h>
#include "Arduino.h"
#include <grideye.h>
#include <GE_SoftUart.h>
#include <GE_SoftUartParse.h>
/*****************************************************************************
  variable value definition
*****************************************************************************/
static uint16_t Main_Delay = 85;
       grideye  GE_GridEyeSensor;
       uint8_t  aucThsBuf[2];            /* NTC thermistor temperature value */
       short    g_ashRawTemp[64];        /* Temperature values of 64 pixels  */
       char buf[2];


/****************************************************************************
  Function：GE_SentDatatoPC
  Description：Used to Sent data to PC
  Input：None
  Output：None
  Others：None
****************************************************************************/
void GE_SentDataToPC( void )
{

}

/****************************************************************************
  Function：GE_SentDatatoPC
  Description：Used to Sent data to PC
  Input：None
  Output：None
  Others：None
****************************************************************************/
void GE_SentDataToPhone( void )
{
```

```
25.00 24.75 23.50 25.25 26.25 24.25 24.00 23.75
26.00 25.00 23.75 24.25 24.75 25.00 24.75 24.75
25.25 24.00 23.75 24.50 24.25 25.25 24.75 25.25
27.00 25.00 25.50 24.00 24.75 25.50 24.75 26.75
[Measurement end]
[Average temperature of all pixels :24C]

[Measurement start]
Raw thermistor data     : HH:1 HL:11010111
Thermistor temperature : [C]: 29.44
24.50 26.00 23.75 25.25 25.75 25.00 23.75 24.25
24.25 23.50 23.75 23.50 23.75 23.50 23.00 24.25
25.75 25.25 24.00 24.00 24.00 23.25 24.50 22.75
25.25 26.50 26.25 24.75 23.75 24.00 23.75 23.50
23.75 24.25 24.50 26.75 25.50 24.25 23.50 24.75
24.25 23.25 23.75 24.75 24.25 25.25 25.25 25.50
26.75 24.00 24.25 23.50 25.00 26.00 24.00 24.75
28.25 26.50 26.00 25.50 24.75 23.75 24.50 25.75
[Measurement end]
[Average temperature of all pixels :24C]

[Measurement start]
Raw thermistor data     : HH:1 HL:11010111
Thermistor temperature : [C]: 29.44
24.50 25.25 24.00 25.00 26.50 23.75 23.25 23.00
24.50 24.25 23.25 24.00 24.00 24.25 22.75 24.00
25.25 25.00 23.50 23.25 23.25 23.25 23.00 23.00
25.75 26.00 26.00 24.00 24.00 22.75 24.25 24.25
24.50 24.50 24.00 24.75 25.75 23.00 24.25 24.25
26.00 24.50 24.25 23.25 24.00 25.50 24.75 25.25
26.25 23.75 24.00 23.75 24.50 25.25 24.50 25.00
27.00 25.50 24.50 24.75 24.50 25.25 24.75 25.25
[Measurement end]
[Average temperature of all pixels :24C]
```

☑ Autoscroll　Both NL & CR　57600 baud

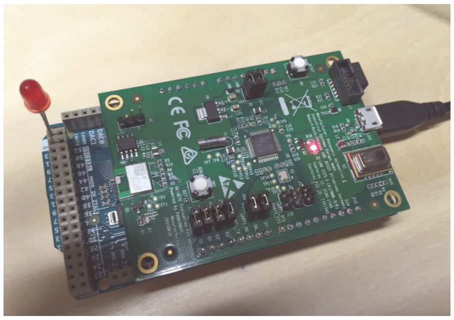
5| Arduino C code to read out pixels' and NTC thermal data

In case the average temperature value of the 64 pixels exceed the preset threshold value (set by variable temperature_treshold = 25°C), we suppose a heat source (the human hand in our case) is present in front of the sensor. (No need motion, presence is enough, this is the biggest advantage of this technology in comparison with doppler radar sensor or PIR sensor technology). Further enhancement could be realized if only those pixels take part in the average creation, whose temperature value exceeds the temperature measured by the NTC on board of GridEye.

In case of detecting gesture, we use one of the available GPIO port of Arduino Due to set its value to logical “HIGH” (D53 in the sample program). During debugging a simple LED (and a serial current limiter resistor) could be placed between GND and D53 to verify adequate operation.

We mentioned before, that Arduino DUE’s GPIOs provide maximum 3.3V signal level when active, which is not enough usually to operate a popular 5V relay module directly.

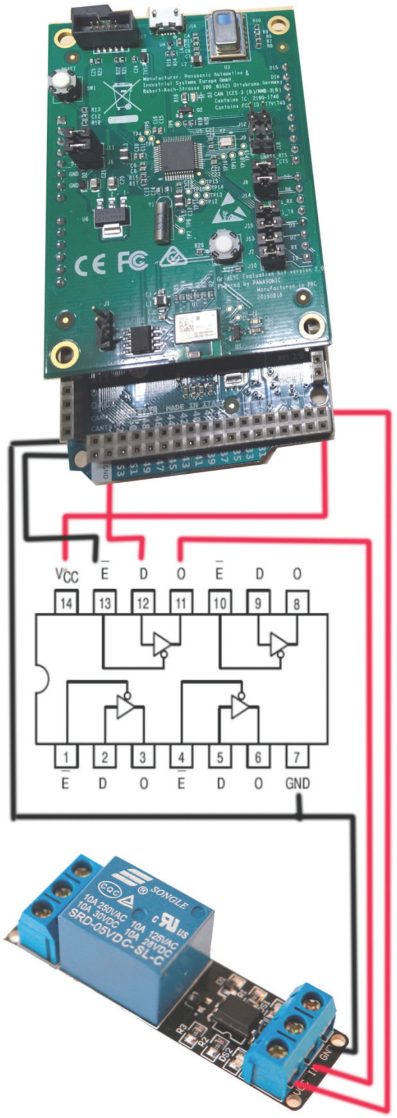
In order to match voltage levels, an external circuit is required, such as SN74HC125N logic level converter IC, which contains 4 channels in a 14 pin DIP package.



6| Arduino DEU with GridEye shield - testing operation

For driving a standalone relay, we need only one D_{OUT}, as well as only one channel of the logic level converter, and usually the power provided by the digital out port (max 40 mA) is enough to operate it. However, should we use more relays, we need to drive their coils with external power supply.

We should never forget about bringing the GND of the power supply to the same potential as the Arduino GND and it is also a good idea to use an overvoltage protection diode. With the described solution we used the GridEye sensor as a presence detector and a gesture controller, that can operate even a line voltage circuit by gestures.



7| Logic level matching, and relay control from the ARDUINO

Using GridEye evaluation kit with PC

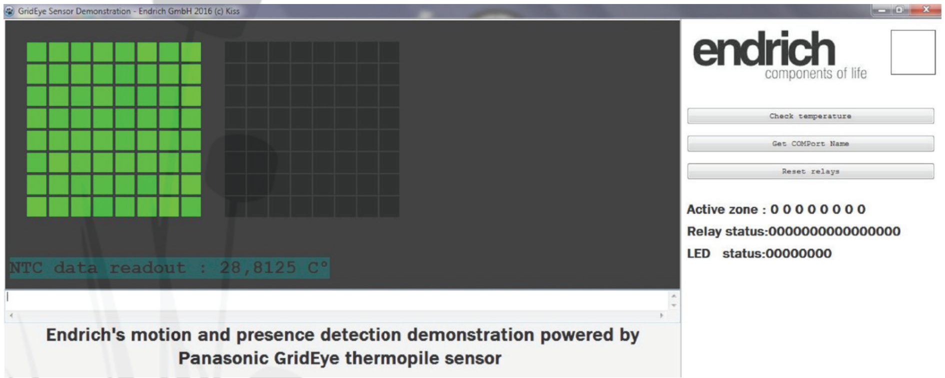
The GridEye Evaluation kit can also be used with a PC connected via USB interface. Using the provided device driver, the USB port can be accessed from the PC as a virtual serial (COM) port. The data structure is the same, user can choose any convenient development system to write the code to convert raw sensor data to visual thermal result. As mentioned there is a free software provide by Panasonic, but we also can make out own version.

I have chosen Lazarus as the development system, being a GNU licensed free software, very much like the popular Pascal powered Delphi system or Borland C++ Builder. It is not my intention to detail the software solution, I would rather give a short overview.

The application made by this way is a lamp control, that follows the motion of a heat source by lighting up lamps one after the other. The pixels detecting higher temperature than the ambient temperature will be highlighted. Each column is connected to a relay, if the hot-spot moves through the column, its lamp will light it up. In this way we can illuminate the moving object, we may “follow” the motion by light.

Examples are available on the screen shots.

The block diagram of the whole system can be found on Fig. 10. The light produced by the 8 vertical high-power LED stripes follow the object moving in front of the sensor, which can be a hand (gesture control) or people walking right and left.

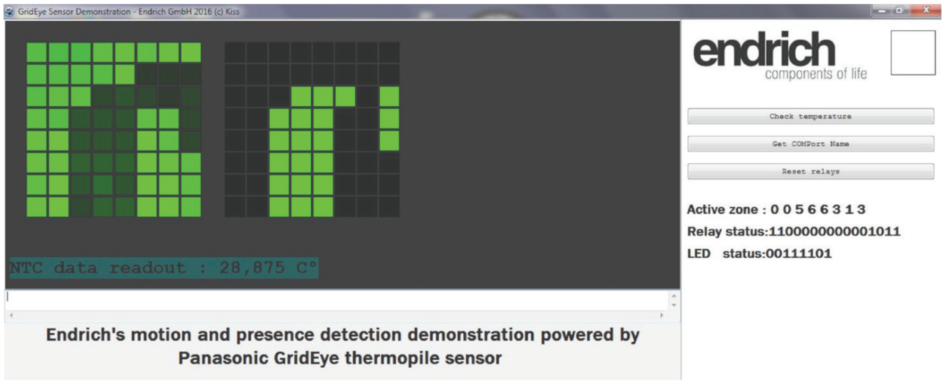


NTC data readout : 28,8125 C°

Active zone : 0 0 0 0 0 0 0 0
Relay status:0000000000000000
LED status:00000000

Endrich's motion and presence detection demonstration powered by Panasonic GridEye thermopile sensor

8| Nothing is in front of the sensor



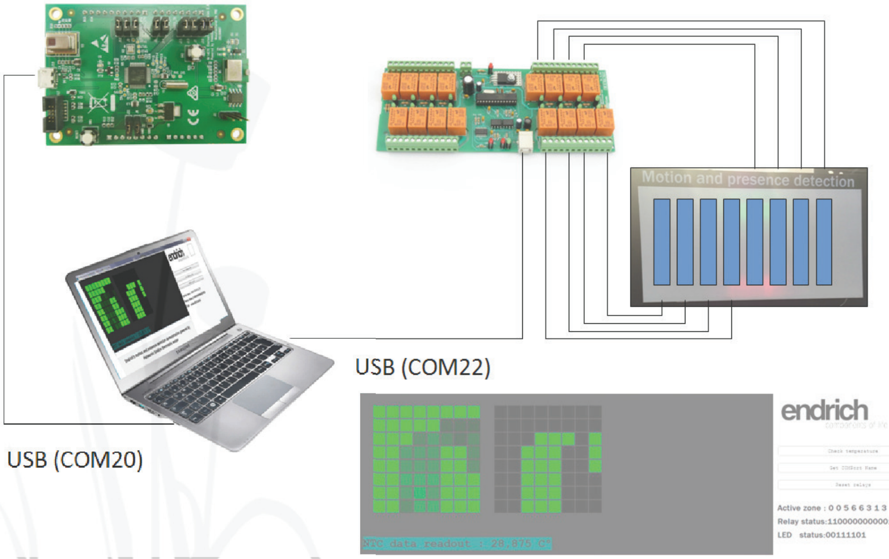
Active zone : 0 0 5 6 6 3 1 3
Relay status:1100000000001011
LED status:00111101

NTC data readout : 28,875 C°

Endrich's motion and presence detection demonstration powered by Panasonic GridEye thermopile sensor

9| Hand in front of sensor (left side: simplified thermal image – right side: target object highlighted)

The third (closing) part of our article series will be about a standalone Arduino Due project, that will display the thermal image detected by the GridEye sensor on an 8X8 high power RGB LED matrix, while we will follow the motion of the heat source with a servo motor. This may be a simulation of a project of an automatic stage lamp that follows the movement of the actor on stage.


Motion and presence detection
USB (COM22)
USB (COM20)
endrich
Active zone : 0 0 5 6 6 3 1 3
Relay status:1100000000001011
LED status:00111101
NTC data readout : 28,875 C°

10| Block diagram of PC based movement detection