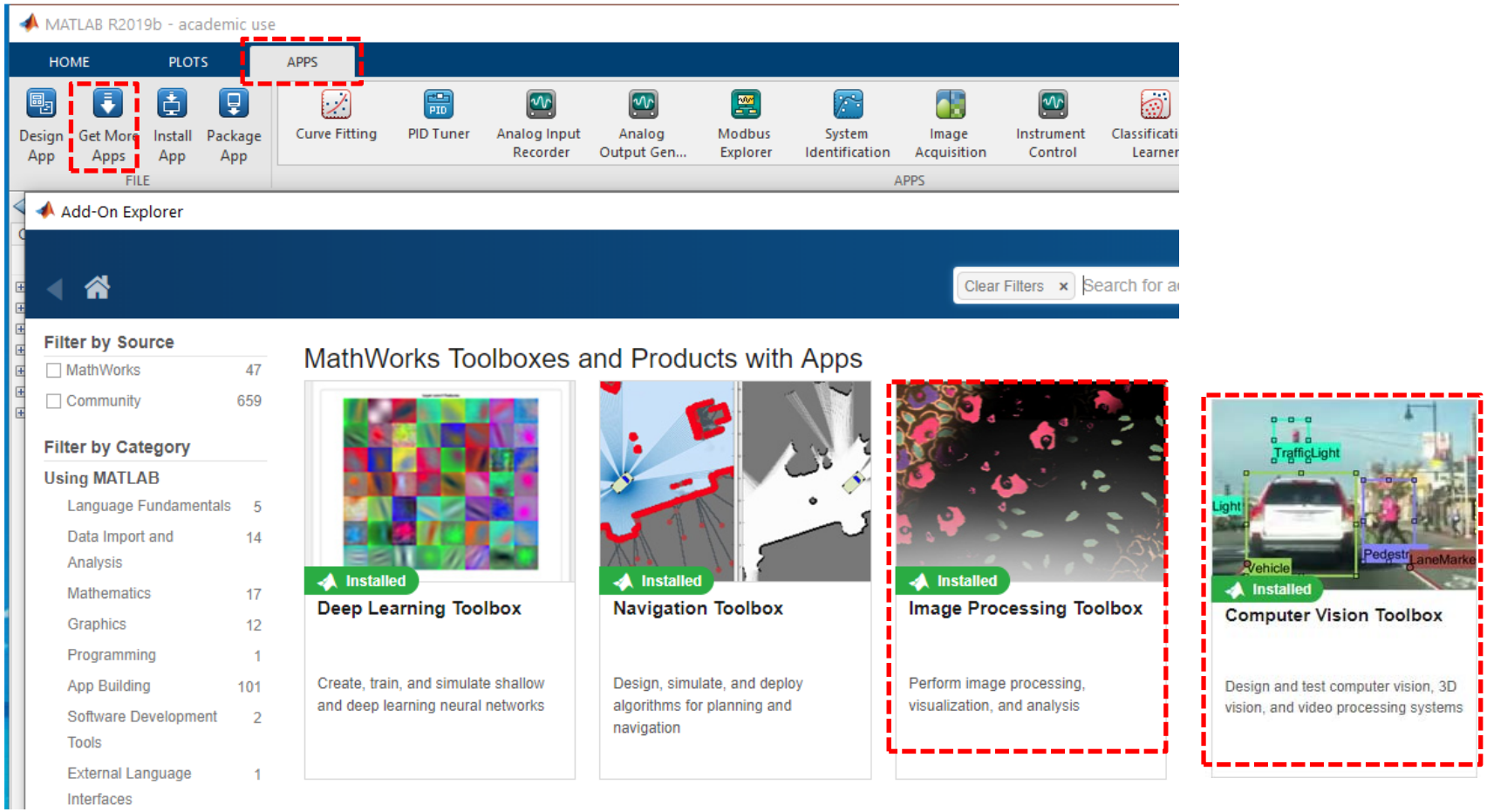

# ACTIVE AND ASSISTED LIVING

## 2022 Winter Semester

# MATLAB Toolbox Installation



# Image Processing Stages

- **Image processing stage**
  - 1. Acquisition
  - 2. Pre-processing
  - 3. Feature extraction and classification(recognition)

# Image Processing Stages

- **Image processing stage**
  - **1. Acquisition**
  - 2. Pre-processing
  - 3. Feature extraction and classification(recognition)

# 1. Image Acquisition

- **Different types of cameras and images**
  - Place of installation – outdoors, indoors
  - Mechanical capacities – bullet-type or pan-tilt-zoom cameras
  - In-built features such as motion detection or night vision
  - Omnidirectional cameras - increased field of view

Bullet-type camera

Pan-tilt-zoom camera

Image from a night vision camera

Image from an omnidirectional camera

Immersive Media's Dodeca 2360 camera

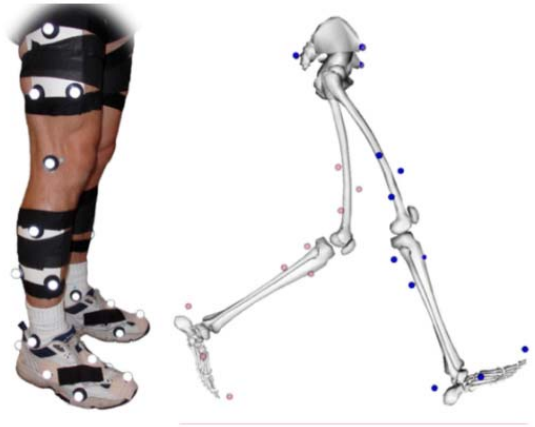
# 1. Image Acquisition

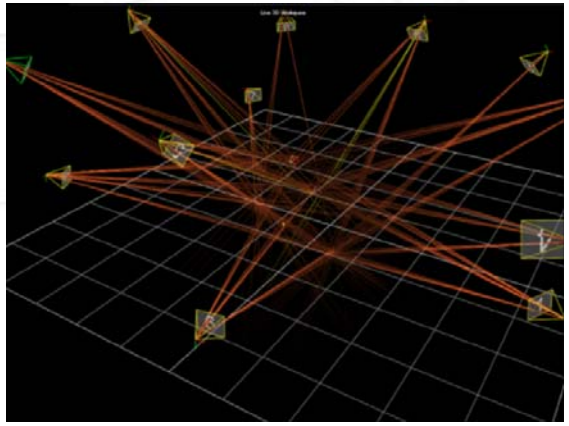
- **Limitation of fixed cameras**
  - Limited field of view
  - Occlusions – difficulties to keep all body parts visible
  - -> wearable cameras : e.g. GoPro or Google Glass

Image from a wearable camera

- **Additional to RGB cameras**
  - Depth cameras, based either on time-of-flight (TOF) or structured light
    - Markerless human body pose estimation

# 1. Image Acquisition





Marker-based motion capture system

https://simtk-confluence.stanford.edu/display/OpenSim/Collecting+Experimental+Data

- **Additional to RGB cameras**
  - Depth cameras, based either on time-of-flight (TOF) or structured light
    - Markerless human body pose estimation
  - Thermal camera
    - Infrared radiation of the scene
    - Person segmentation and pose estimation

# How to Obtain and Display Images / Video

- **MATLAB Functions**
  - imread

    Read image from graphics file

    gif, png, tiff, jpg, bmp, etc.

| | | |
|---|---|---|
| BMP — Windows Bitmap | JPEG — Joint Photographic Experts Group | PNG — Portable Network Graphics |
| CUR — Cursor File | JPEG 2000 — Joint Photographic Experts Group 2000 | PPM — Portable Pixmap |
| GIF — Graphics Interchange Format | PBM — Portable Bitmap | RAS — Sun Raster |
| HDF4 — Hierarchical Data Format | PCX — Windows Paintbrush | TIFF — Tagged Image File Format |
| ICO — Icon File | PGM — Portable Graymap | XWD — X Window Dump |

  - image

    Display image from array

  imshow

  Display image

# How to Obtain and Display Images/Video

- **Graphics File – local storage**
- **Acquire Images from Webcams**
- **Remote Cameras**

# How to Obtain and Display Images / Video

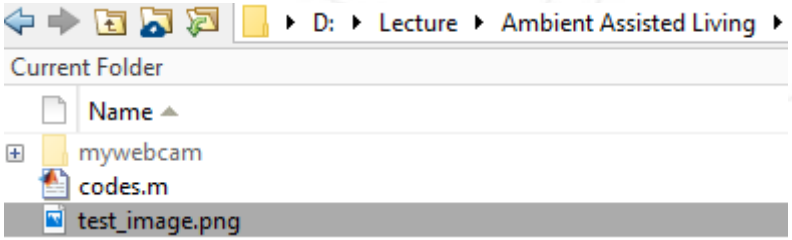
- **Graphics File**

  C:\Program Files\MATLAB\R2021b\toolbox\images\imdata

- **Acquire Images from Webcams**

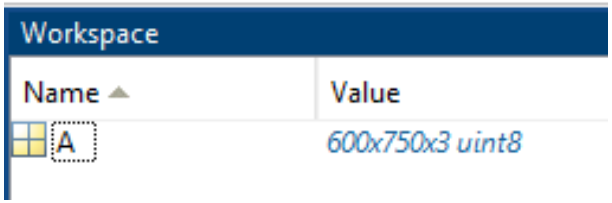
- **Remote Cameras**

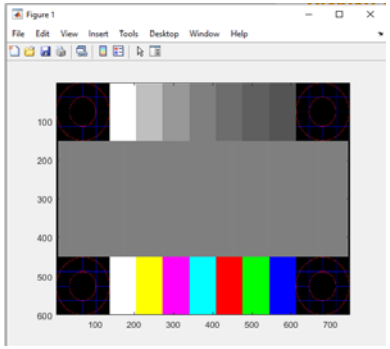
# Display Images




```
>> A = imread('test_image.png');
figure(1)
image(A)
```

A = imread('D:\Lecture\AAL\MATLAB\test_image.png');



# How to Obtain and Display Images/Video

- **Graphics File – local storage**
- **Acquire Images from Webcams**
- **Remote Cameras**

# How to Obtain and Display Images / Video

- **Acquire Images from Webcams**

```
>> webcamlist
Error using webcamlist (line 20)
MATLAB Support Package for USB Webcams has not been installed. Open Add-On Explorer to install the Webcam Support Package.
```

**MATLAB Support Package for USB Webcams**

by MathWorks Image Acquisition Toolbox Team STAFF

Acquire images and video from UVC compliant webcams.

Install

Hardware Support

## Webcam Image Acquisition

Acquire images from webcams

### Functions

| | |
|---|---|
| webcamlist | List of webcams connected to your system |
| webcam | Connection to a webcam |
| preview | Preview live video data from webcam |
| snapshot | Acquire single image frame from a webcam |
| closePreview | Close webcam preview window |

## preview

Preview live video data from webcam

# WebCam

```
>> webcamlist

ans =

  1×1 cell array

    {'NEC HD WebCam'}
```

```
>> cam = webcam(1)

cam =

  webcam with properties:

                    Name: 'NEC HD WebCam'
    AvailableResolutions: {'1280x720'  '640x480'  '640x360'  '320x240'}
              Resolution: '1280x720'
       WhiteBalanceMode: 'auto'
              Sharpness: 50
                    Hue: 0
             Brightness: 0
    BacklightCompensation: 0
                  Gamma: 300
           WhiteBalance: 4600
             Saturation: 64
               Contrast: 50
```

OR

```
cam = webcam('NEC HD WebCam')
```

```
>> preview(cam);
```

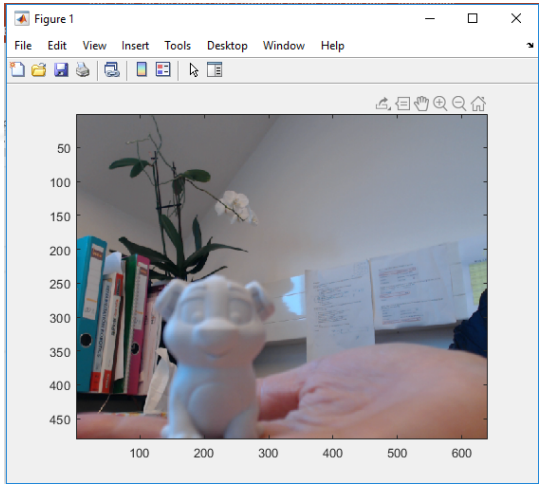
# WebCam

## snapshot

Acquire single image frame from a webcam

cam=webcam('HD Pro Webcam C920')

```
>> img=snapshot(cam);
>> image(img)
```




## Multiple snapshots

```
>> for i = 1:3
    img = snapshot(cam);
    figure
    image(img);
end
```

clear cam

# Capturing a Video from a Webcam

MANAGEMENT CENTER INNSBRUCK

## VideoWriter

Create object to write video files

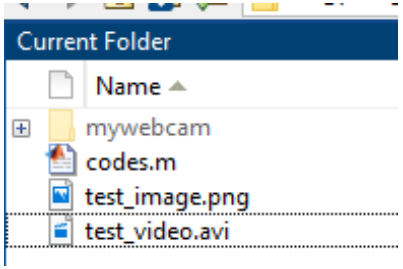
## open

Open file in appropriate application

## writeVideo

Write video data to file

```
video_out = VideoWriter('test_video.avi');
open(video_out);

for index = 1:30
    % Acquire frame for processing
    img = snapshot(cam);
    % Write frame to video
    writeVideo(video_out,img);
end

close(video_out);
clear cam
```



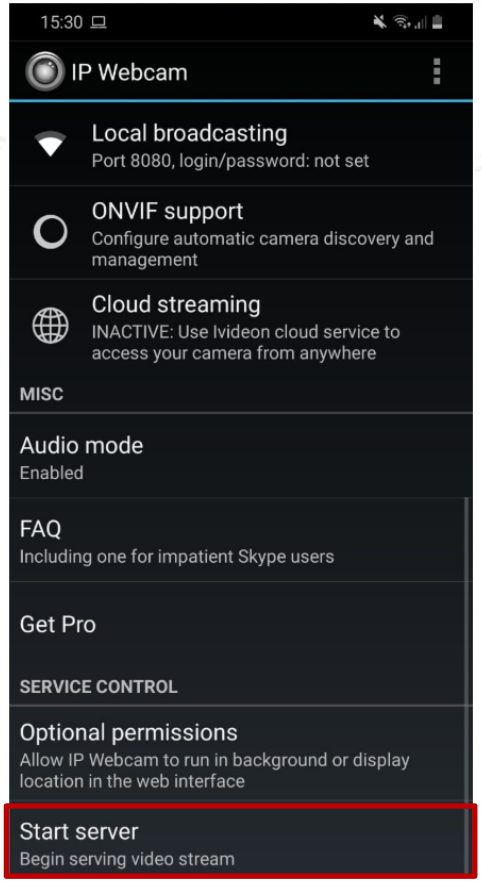
# How to Obtain and Display Images/Video

- **Graphics File – local storage**
- **Acquire Images from Webcams**
- **Remote Cameras**

# How to Obtain Images / Video

Android


IP Webcam App





IP address

iPhone

IP Cam

# How to Obtain Images/Video

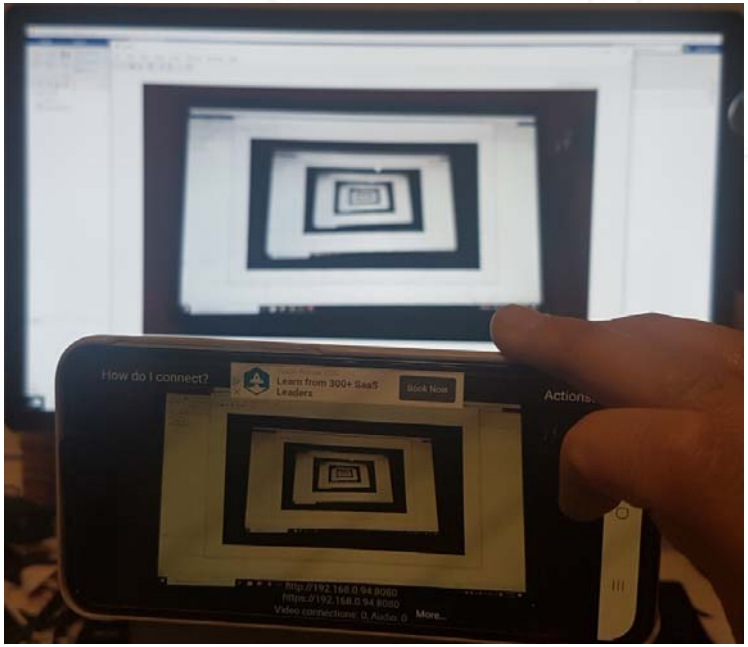
http://192.168.0.94:8080/video

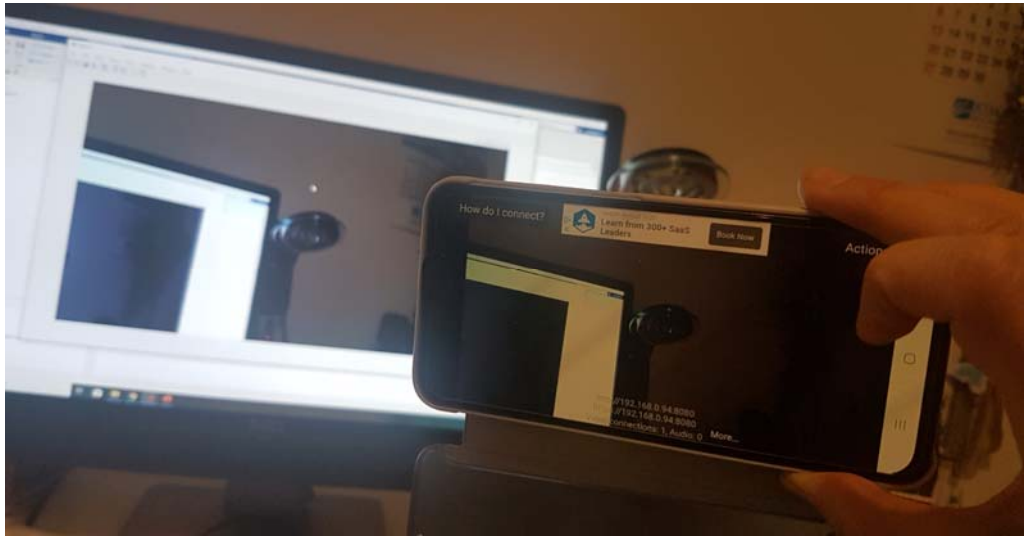
Editor - mywebcam.m*

mywebcam.m*

```
1 -    webcam_url = 'http://192.168.0.94:8080/shot.jpg';  %shot.jpg the last scene
2
3
4 -    Scene = imread(webcam_url);
5 -    figure(1);
6 -    hC= imshow(Scene);
7
8 -    while 1
9 -      Scene = imread(webcam_url);
10 -     set(hC,'CData',Scene)
11 -     drawnow;
12 -   end
```

# IP Cam Display



# Image Processing Stages

- **Image processing stage**
  - 1. Acquisition
  - **2. Pre-processing**
  - 3. Feature extraction and classification(recognition)

# 2. Image Pre-processing

- Main focus: recording persons, detect human silhouette – the **region of interest (ROI)**

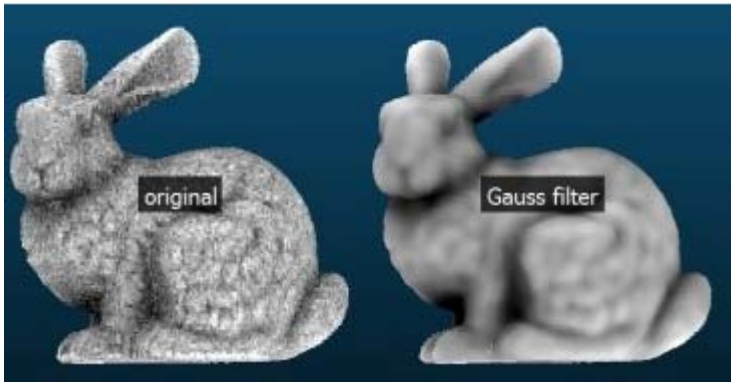
- 1) Separating the ROI (motion segmentation) from the background (static)

- 2) Processing: normalizing pixels, dimension reduction to apply pattern recognition techniques or machine learning algorithms

# 2. Image Pre-processing

- **Dimension reduction  to ensure stable real-time execution**
  - Downsampled (e.g. 15 fps)
  - Conversion to 8-bit greyscale images

- **Applying filters (e.g. Gaussian, Median, Mean, Erosion filter)**
  - To reduce noise

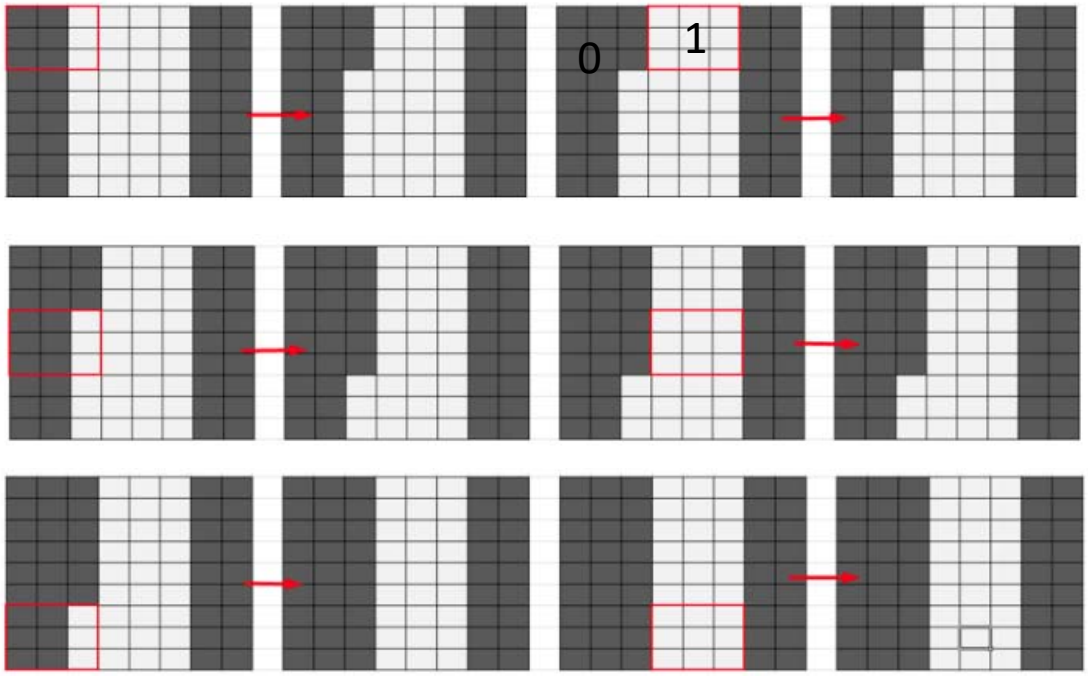




Human silhouettes corresponding to different activities.

# 2. Image Pre-processing

- **e.g.) Erosion filter**




kernal

https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm

# Erosion Filter in MATLAB

```
i= imread('tester2.png'); % load an image
figure(1);  imshow(i); %  display original image

igray= rgb2gray(i);
figure(2);  imshow(igray); %  display gray scale image

ib = im2bw(igray,0.5); % converts the grayscale image  to binary image
figure(3);  imshow(ib); %  display binary image

mask = strel('diamond',7) ; % flat morphological structuring element - line , square, disk
ie = imerode(ib,mask); % Erode image
figure(3); imshow(ie); %  display the result image
clear
```

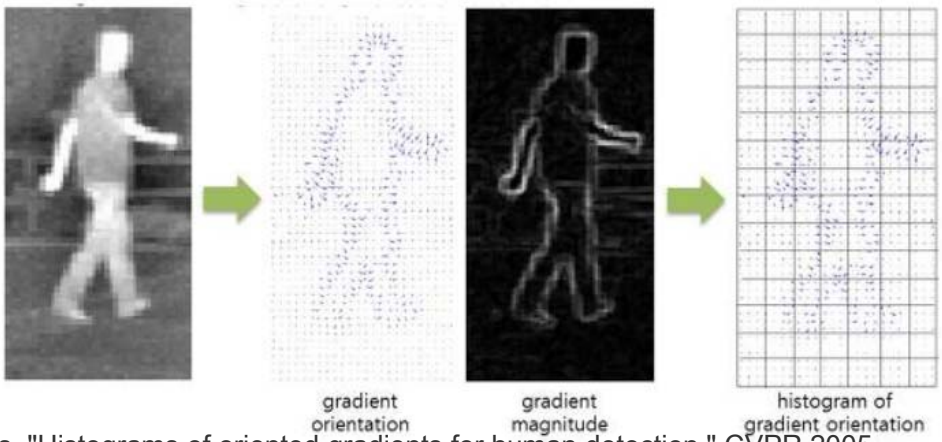
# Erosion Filter

# Image Processing Stages

- **Image processing stage**
  - 1. Acquisition
  - 2. Pre-processing
  - 3. **Feature extraction and classification(recognition)**

# 3. Image Pre-processing Feature Extraction and Recognition

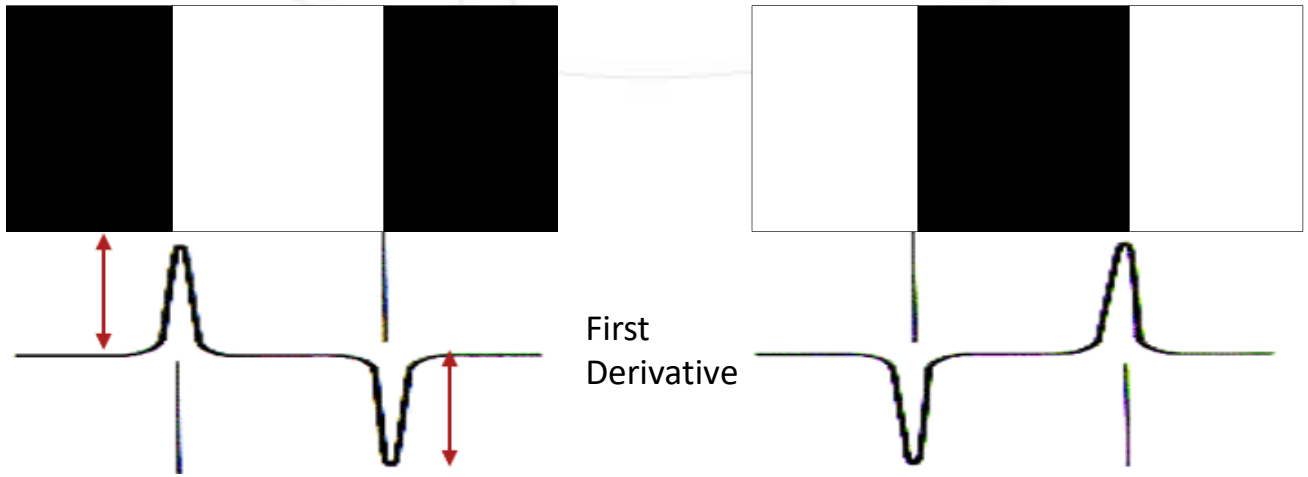
- **Visual features - whole image or on the detected ROIs**
  - To obtain the characteristic information
- **Feature extraction methods**
  - HOGs, SIFT(Scale Invariant Feature Trans form), Haar
- **Histograms of oriented gradients (HOGs)**
  - Very popular feature for human detection

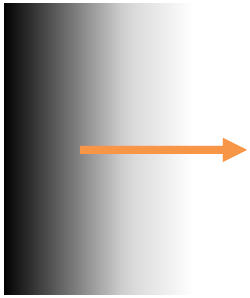



N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," CVPR 2005.

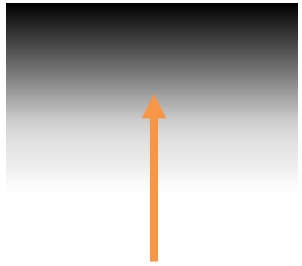
# 3. Image Pre-processing Feature Extraction and Recognition

- **Gradient**



$$\nabla F = \left[\frac{\delta F}{\delta x}, 0\right] = [F_x \ , 0]$$



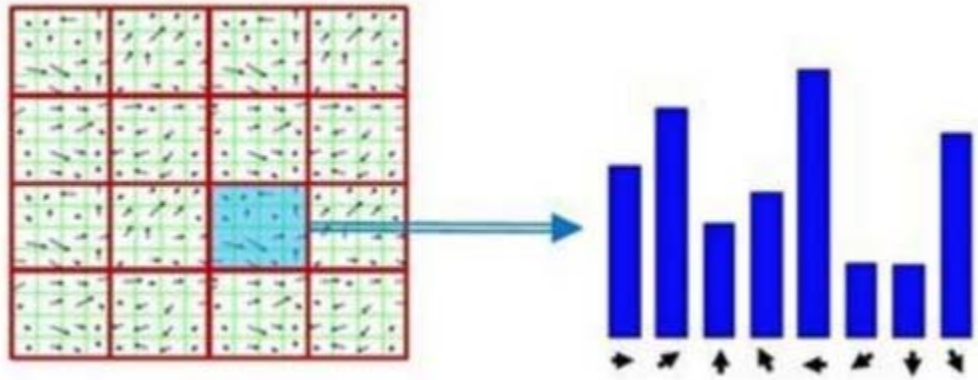
$$\nabla F = \left[0, \frac{\delta F}{\delta y}\right] = [0, F_y \ ]$$

$$Gradient\ Magnitude(G) = \sqrt{F_x^2 + F_y^2}$$

$$Angle(\theta) = tan^{-1}\frac{F_y}{F_x}$$

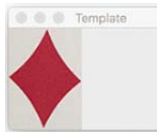
# 3. Image Pre-processing Feature Extraction and Recognition

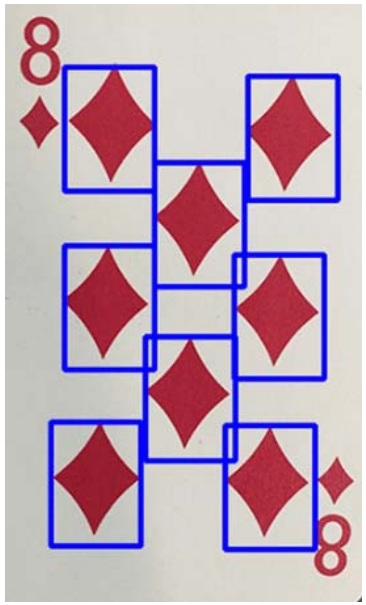
- **Histograms of oriented gradients (HOGs)**

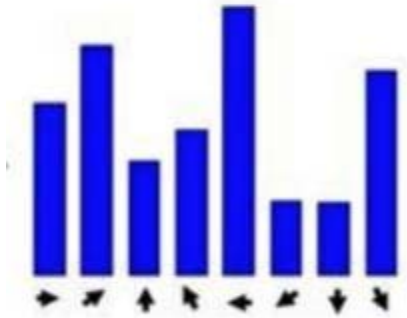


Real-Time Rotation Estimation Using Histograms of Oriented GradientsBlaž Bratanič, Franjo Pernuš, Boštjan Likar, Dejan Tomaževič 2014, PLoS ONE - Article

# 3. Image Pre-processing Feature Extraction and Recognition

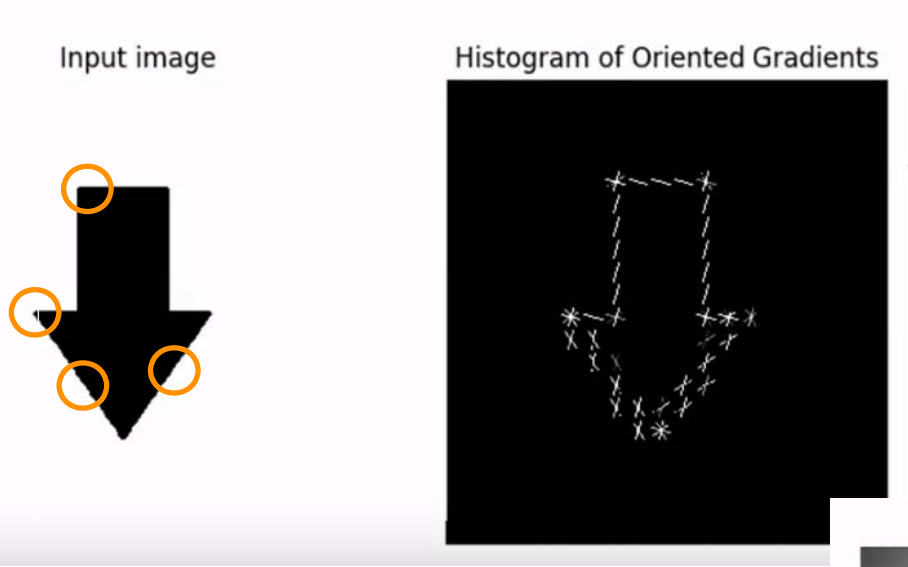


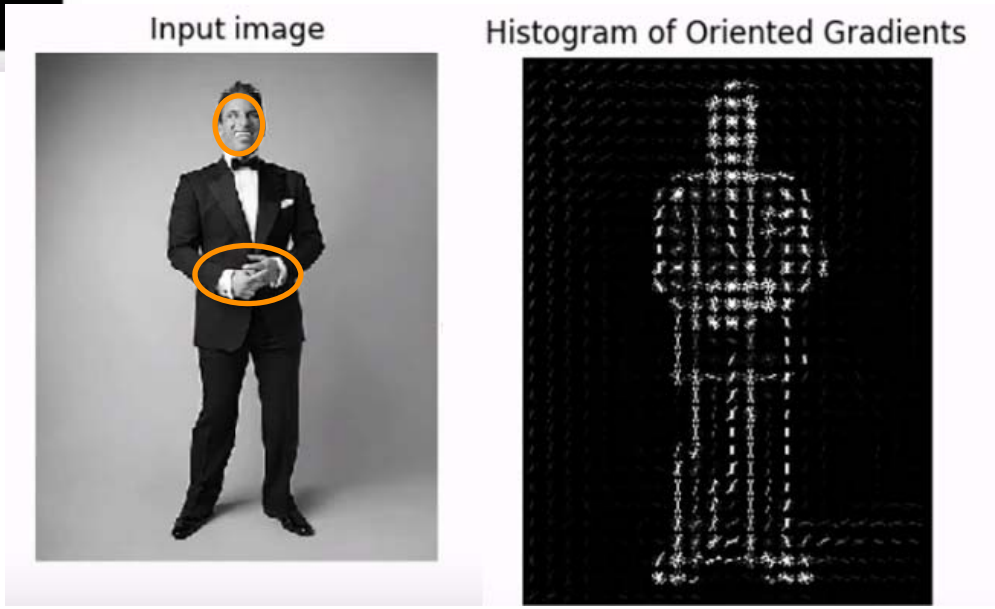





Feature detection Vs. Hogs

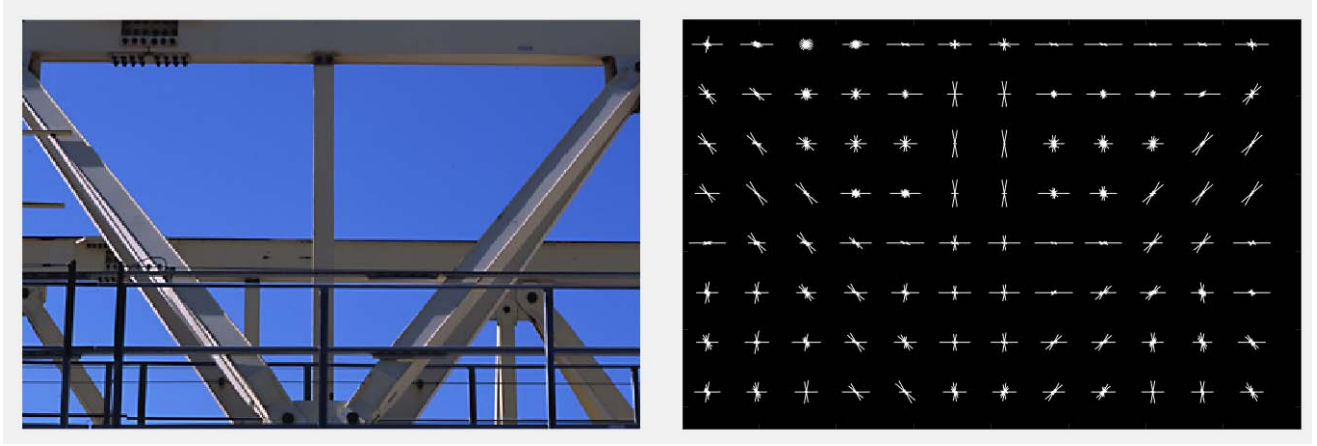
# Histograms of Oriented Gradients (HOGs)



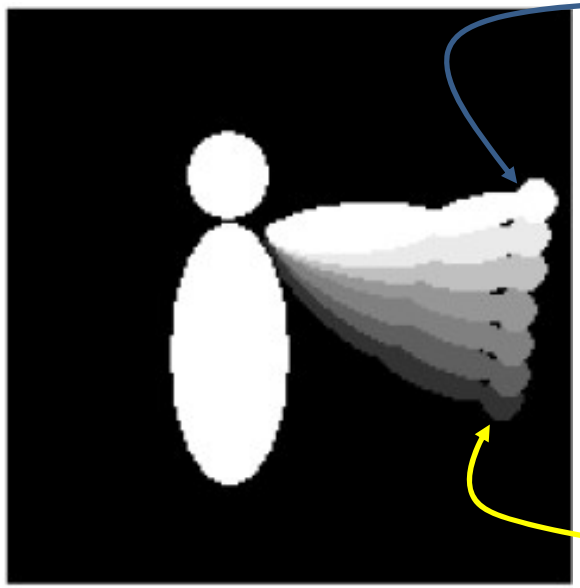
# Histograms of Oriented Gradients (HOGs) in MATLAB

```
img = imread('gantrycrane.png');
[hog1,hogVisualization] = extractHOGFeatures(img,'CellSize',[32 32]);
subplot(1,2,1);
imshow(img);
subplot(1,2,2);
plot(hogVisualization);
```



# 3. Feature Extraction and Recognition

- **Recognition**
  - Motion History Images (HMI)



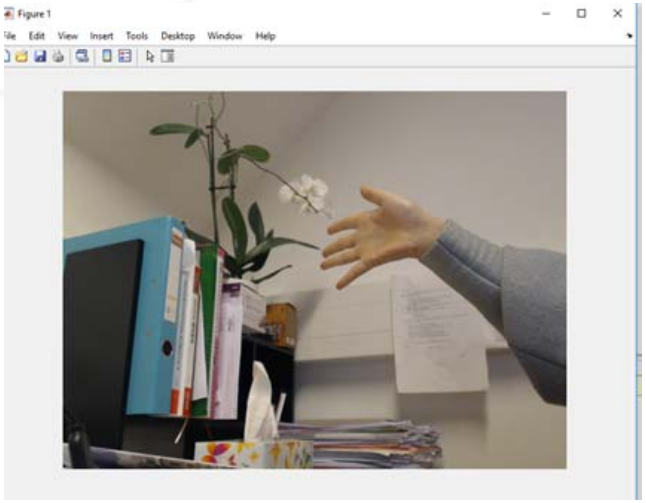

$$\text{if } B_t(x,y) = 1 \quad MHI_t(x,y) := \tau$$

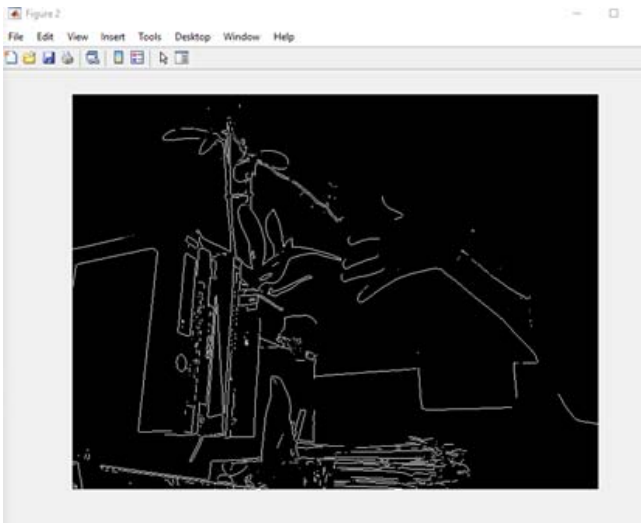
$$\text{else if } MHI_{t-1} \neq 0 \quad MHI_t(x,y) := MHI_{t-1}(x,y) - 1$$

https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch14/ch14_motion_representation.htm

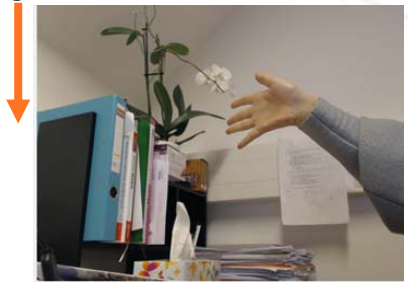
# Image Processing

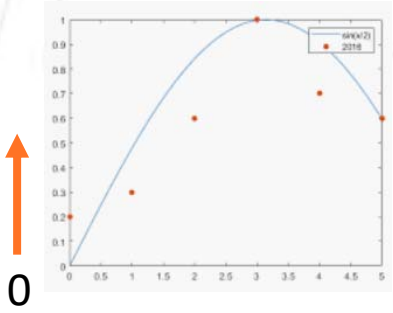
```
clear all;  clc;
cam = webcam('HD Pro Webcam C920')

C= snapshot(cam);
figure(1);
hC = imshow(C)
G=rgb2gray(C);
figure(2);
hG = imshow(edge(G))

while 1

C = snapshot(cam);
G = rgb2gray(C);
set(hC,'CData',C); % set graphic object properties
set(hG,'CData',edge(G));
drawnow;

end
```



# Displaying Images in the Graph




## flipud

Flip array up to down

```
A = 10×1          B = 10×1

     1                10
     2                 9
     3                 8
     4                 7
     5                 6
     6                 5
     7                 4
     8                 3
     9                 2
    10                 1
```

## flip

Flip order of elements

## flipdim

Flip array along specified dimension

`flipdim(A,1)` where

```
A =

     1     4
     2     5
     3     6
```

produces

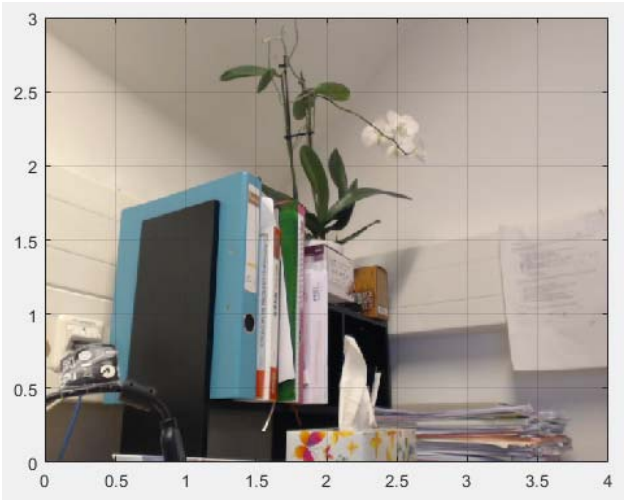
```
     3     6
     2     5
     1     4
```

# Displaying Images in the Graph

```
clear all;  clc;
cam = webcam('HD Pro Webcam C920')

C= snapshot(cam);
figure(1);
hC=imagesc([0 4], [0 3], flipdim(C,1));
% plot up to here first
% Image scale

set(gca, 'YDir', 'normal'); grid on;
```

gca

Current axes or chart

# Color Tracking

- **10 x 10 region**
- **red_region.m**

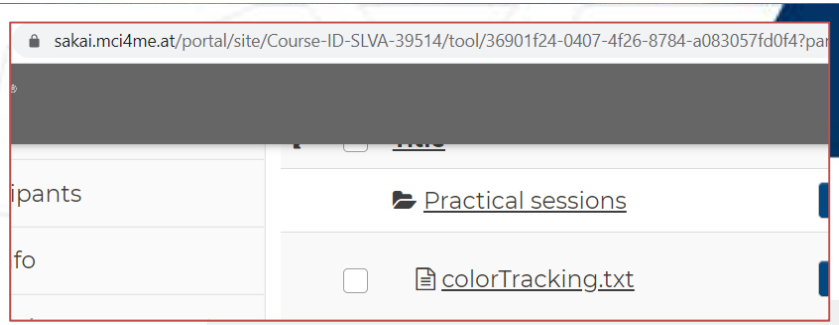
Editor - red_region.m

red_region.m

```
2
3   function [x y] =red_region(C)
4
5   R =  C(:,:,1); G =  C(:,:,2);  B =  C(:,:,3);
6
7   D1=min(R-G, R-B);
8   D = D1 -min(min(D)); %normalize
9   s=zeros(48,64);% 10 x 10 region
10
11  for h=1:48
12
13  i=1+(h-1)*10;
14  j=1+9;
15
16  sr=D(i:j,:);
17  s(h,:)=sum(reshape(sr,[],64));
18  end
19
20  [sMax1 sIdx1] = max(s, [],1);
21
22  [sMax2 sIdx2] = max(sMax1);
23  i=sIdx2;
24  j=49-sIdx1(sIdx1);
25
26  x=i*10;
27  y=j*10;
28
```

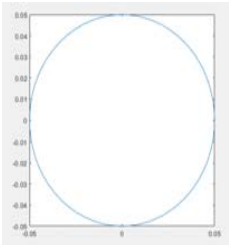
Modify if the resolution of your webcam is different

# Color Tracking

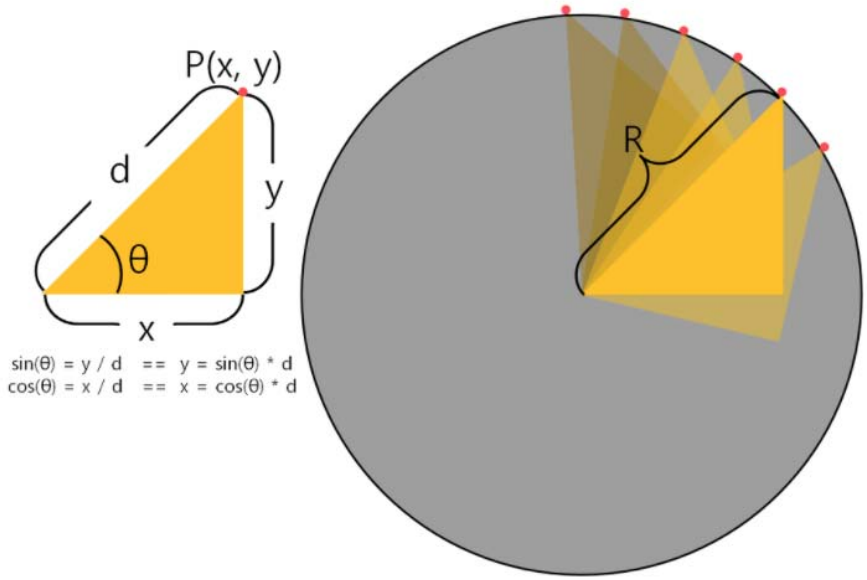



```
clear all;  clc;
cam = webcam('HD Pro Webcam C920')

C= snapshot(cam);
H=size(C,1);
W=size(C,2);
figure(1); % plot up to here first


hC=imagesc([0 4], [0 3], flipdim(C,1));
set(gca, 'YDir', 'normal'); grid on;
t = 0:0.1:2*pi;
x=cos(t); y=sin(t);


while 1
C=snapshot(cam);
set(hC, 'CData', flipdim(C,1));
[rx ry] =red_region(C);
x_pos = 4*rx/W + 0.05*x;
y_pos = 3*ry/H + 0.05*y;

hold on;
plot(x_pos,y_pos,'r');
hold off;

previous_y_pos = y_pos;
pause(0.001);
end
```

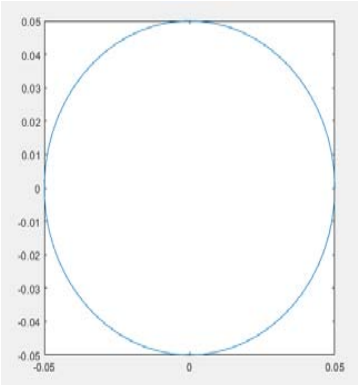

```
>> t = 0:0.1:2*pi;
    x=cos(t); y=sin(t);

    x_pos =  0.05*x;
    y_pos = 0.05*y;
>> plot(x_pos,y_pos)
```



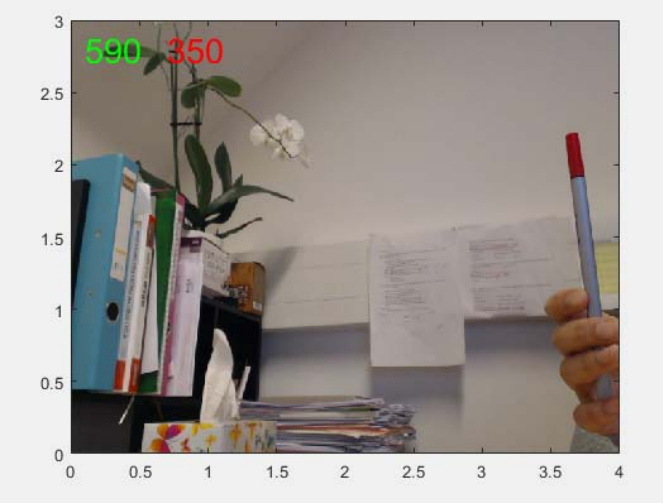
# Drawing a Circle




```
>> t = 0:0.1:2*pi;
     x=cos(t); y=sin(t);

     x_pos =  0.05*x;
     y_pos = 0.05*y;
>> plot(x_pos,y_pos)
```



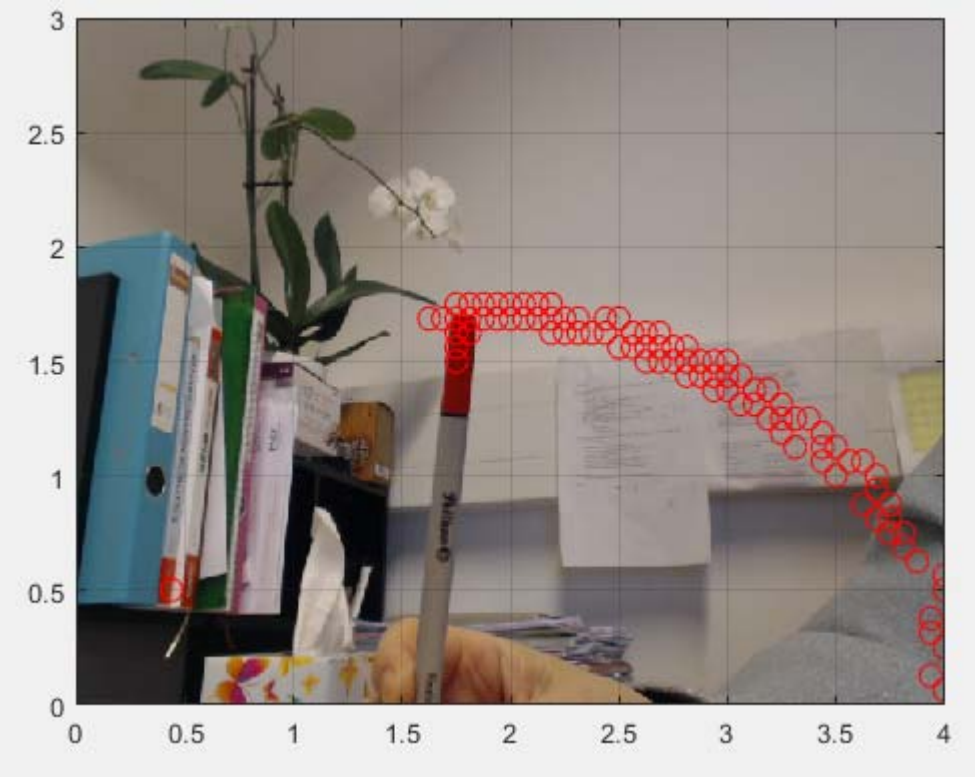
# Color Tracking Center Point Display


```
clear all;  clc;
cam = webcam('HD Pro Webcam C920')

C= snapshot(cam);

figure(1);

hC=imagesc([0 4], [0 3], flipdim(C,1));
set(gca, 'YDir', 'normal'); hold on;

hT3=text(0.1,2.8,'X');
hT4=text(0.7,2.8,'Y');
set(hT3,'Color', [0 1 0]);
set(hT3,'FontSize', 20);
set(hT4,'Color', [1 0 0]);
set(hT4,'FontSize', 20);

while 1

C=snapshot(cam);
set(hC, 'CData', flipdim(C,1));
[rx ry] =red_region(C);
set(hT3,'String', num2str(rx));
set(hT4,'String', num2str(ry));
drawnow;
pause(0.01);
end
```

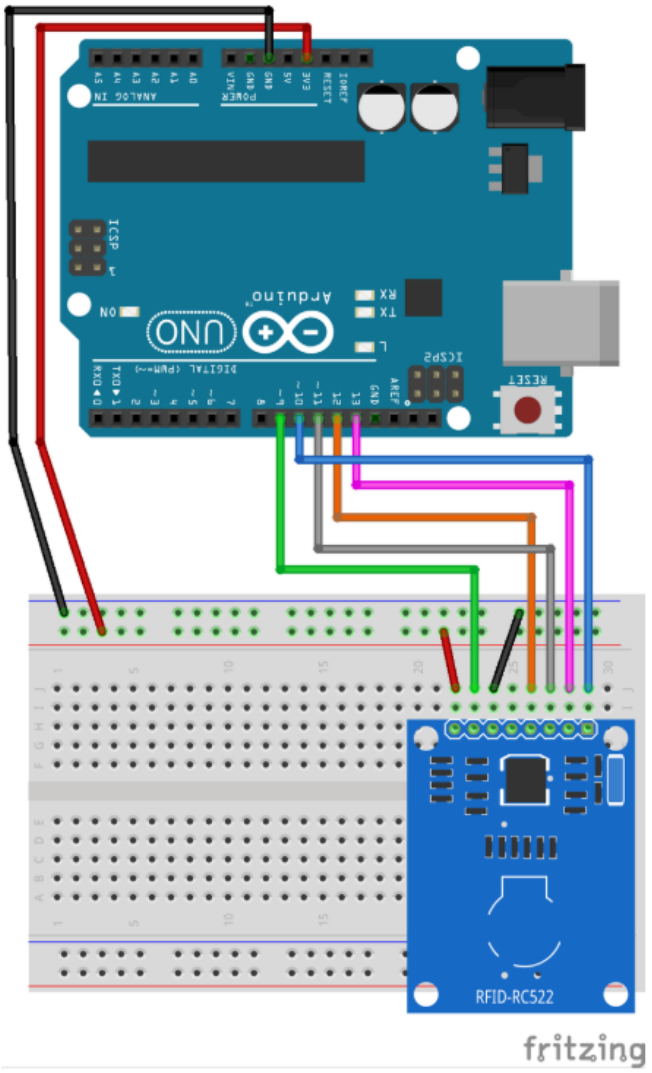
# Fall Detection Alarm

- **Vision + Buzzer**



# RFID

- **RC522 RFID Reader / Writer module**

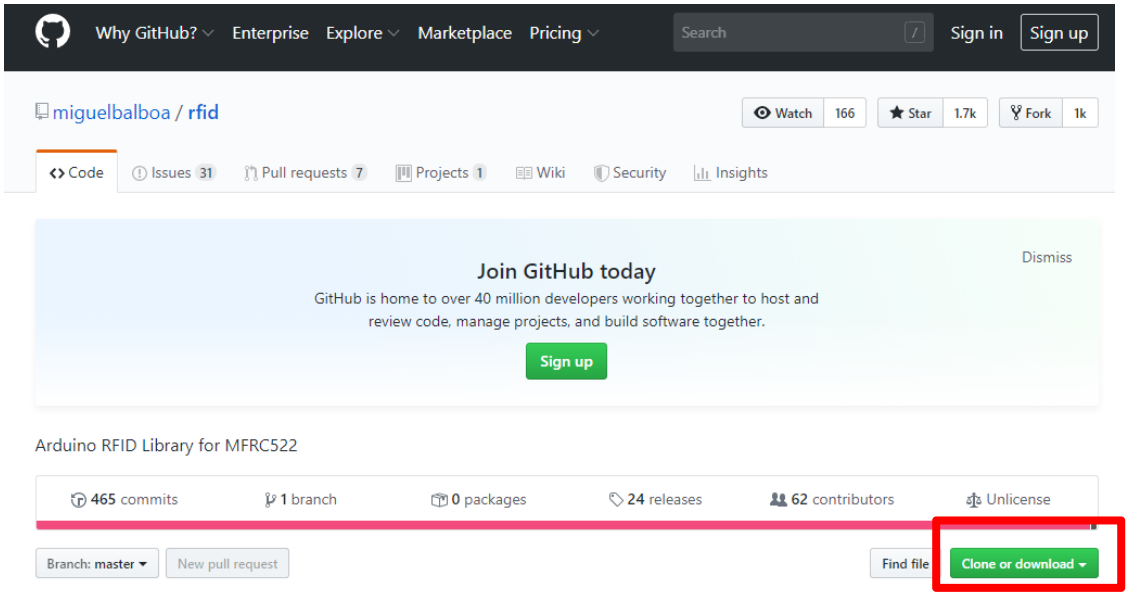



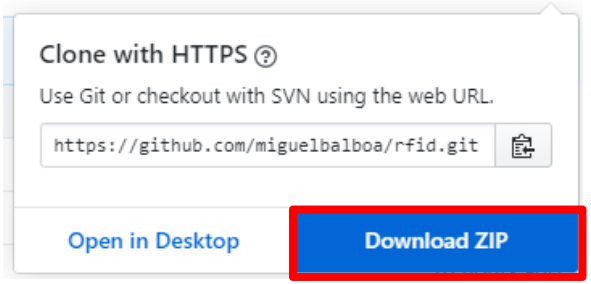

- **Various types of RFID tags**



# RFID

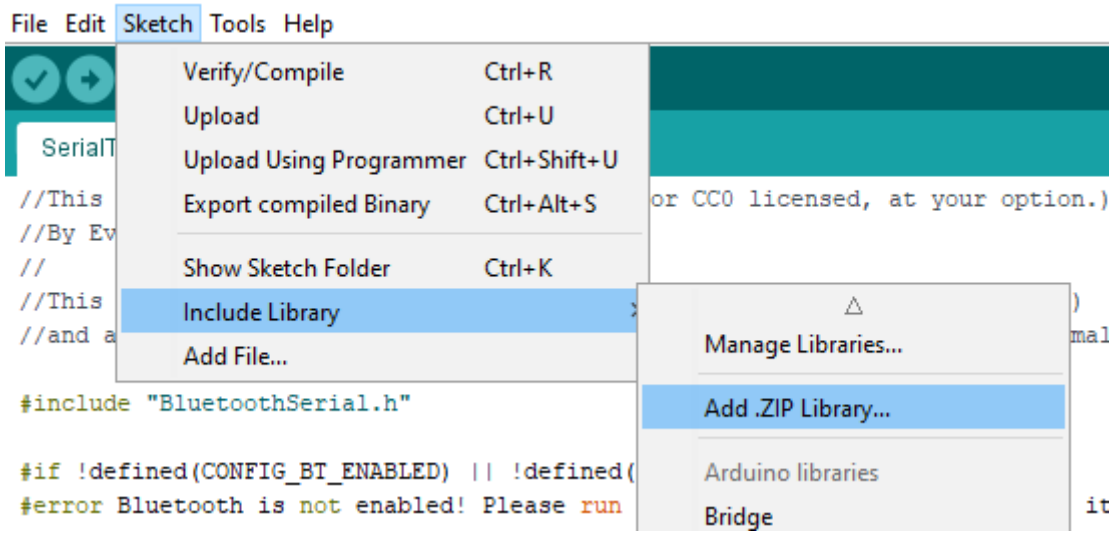
- **RFID library download**
- **https://github.com/miguelbalboa/rfid**



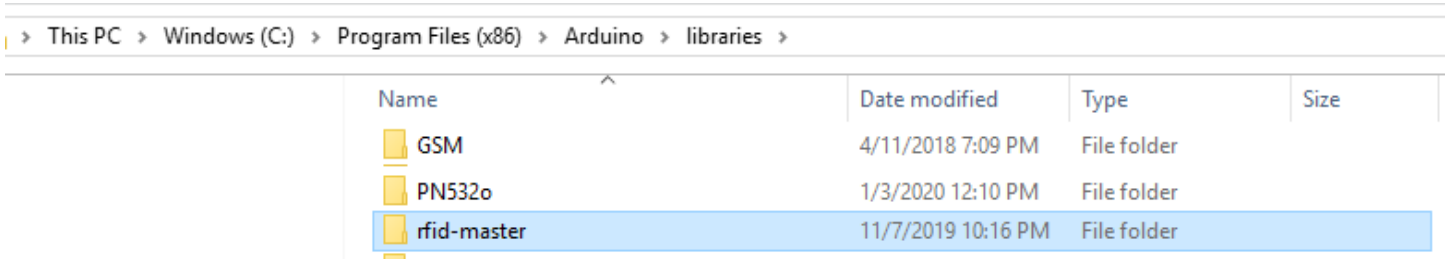
# How to Add Library

1)

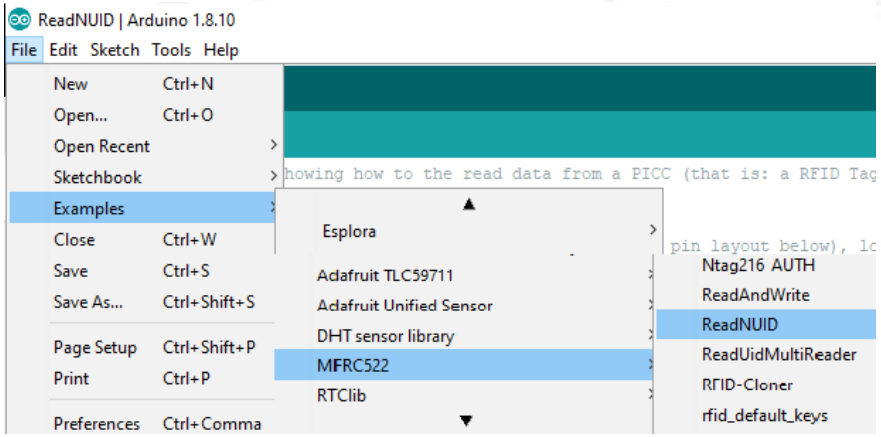


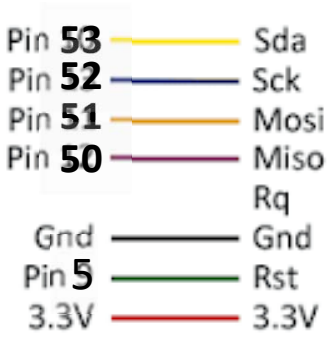

or

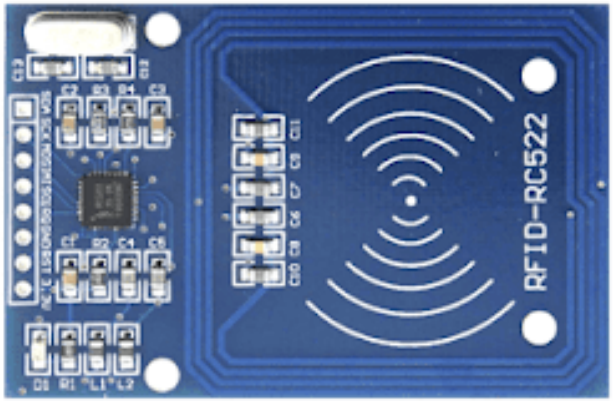
2) Unzip and place it at the following folder



# Read RFID




```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 9
#define RST_PIN 10

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the cla

MFRC522::MIFARE_Key key;

// Init array that will store new NUID
byte nuidPICC[4];
```

```
#define SS_PIN 53
#define RST_PIN 5
```

ReadNUID

```
 * Typical pin layout used:
 * -----------------------------------------------------------------------------
 *             MFRC522      Arduino       Arduino     Arduino     Arduino
 *             Reader/PCD   Uno/101       Mega        Nano v3     Leonardo/M
 * Signal      Pin          Pin           Pin         Pin         Pin
 * -----------------------------------------------------------------------------
 * RST/Reset   RST          9             5           D9          RESET/ICSP
 * SPI SS      SDA(SS)      10            53          D10         10
 * SPI MOSI    MOSI         11 / ICSP-4   51          D11         ICSP-4
 * SPI MISO    MISO         12 / ICSP-1   50          D12         ICSP-1
 * SPI SCK     SCK          13 / ICSP-3   52          D13         ICSP-3
```

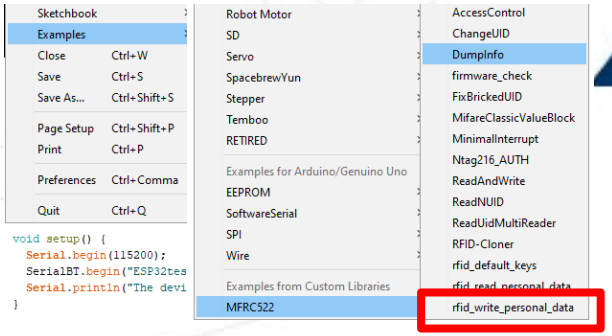
# Read RFID

- **Try to read your student ID (* do not write any data*)**


4 byte UID

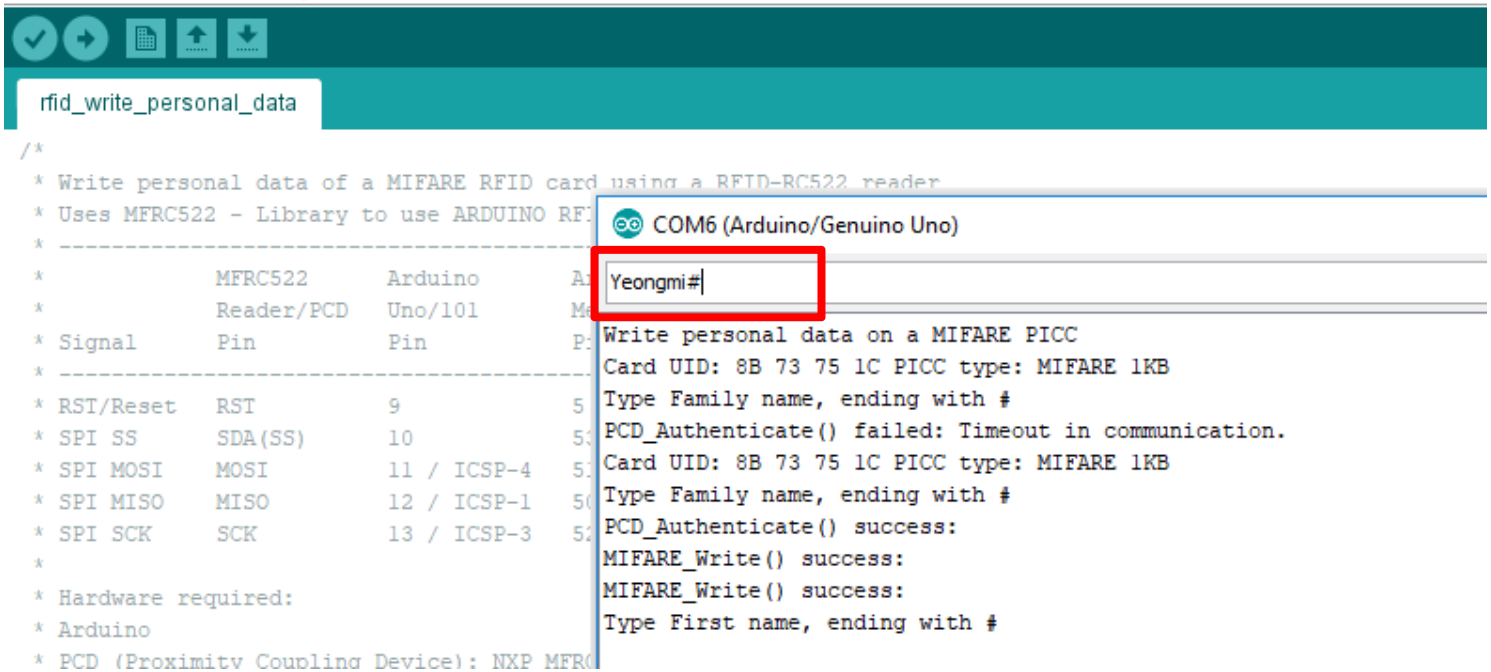
1KB memory

# Write / Read Data

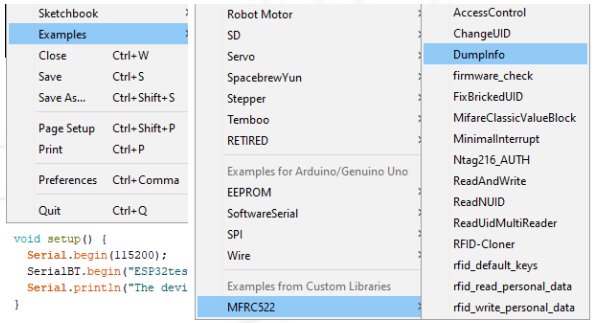



- **Write personal data**

C:\Program Files (x86)\Arduino\libraries\rfid-master\examples\rfid_write_personal_data

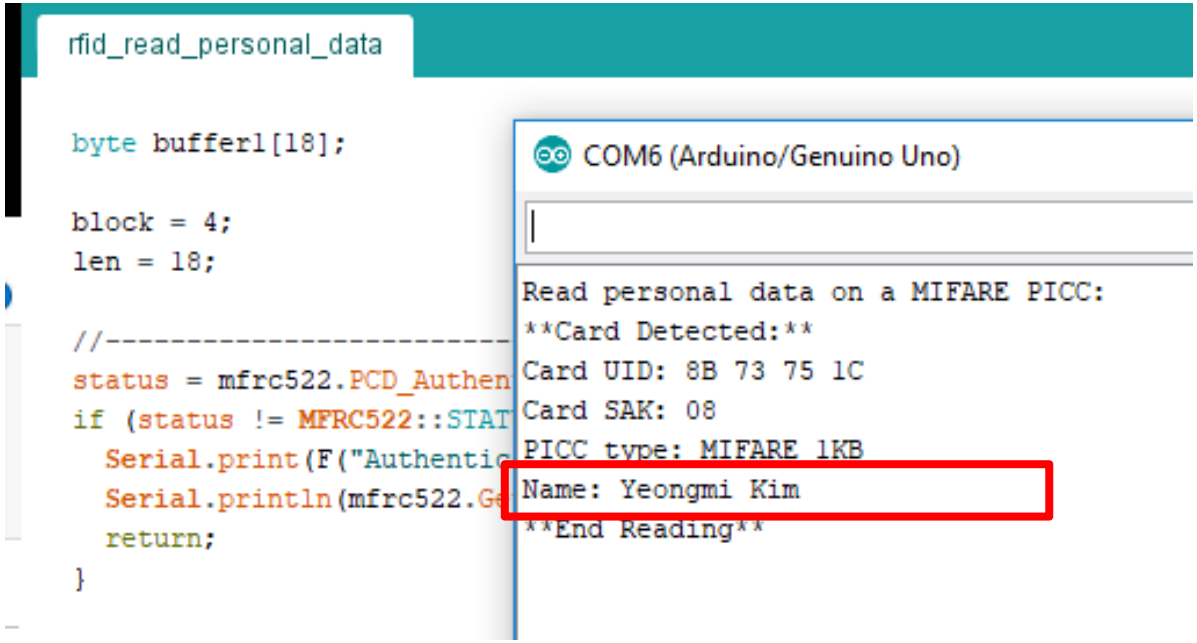


# Write / Read Data




- **Read personal data**

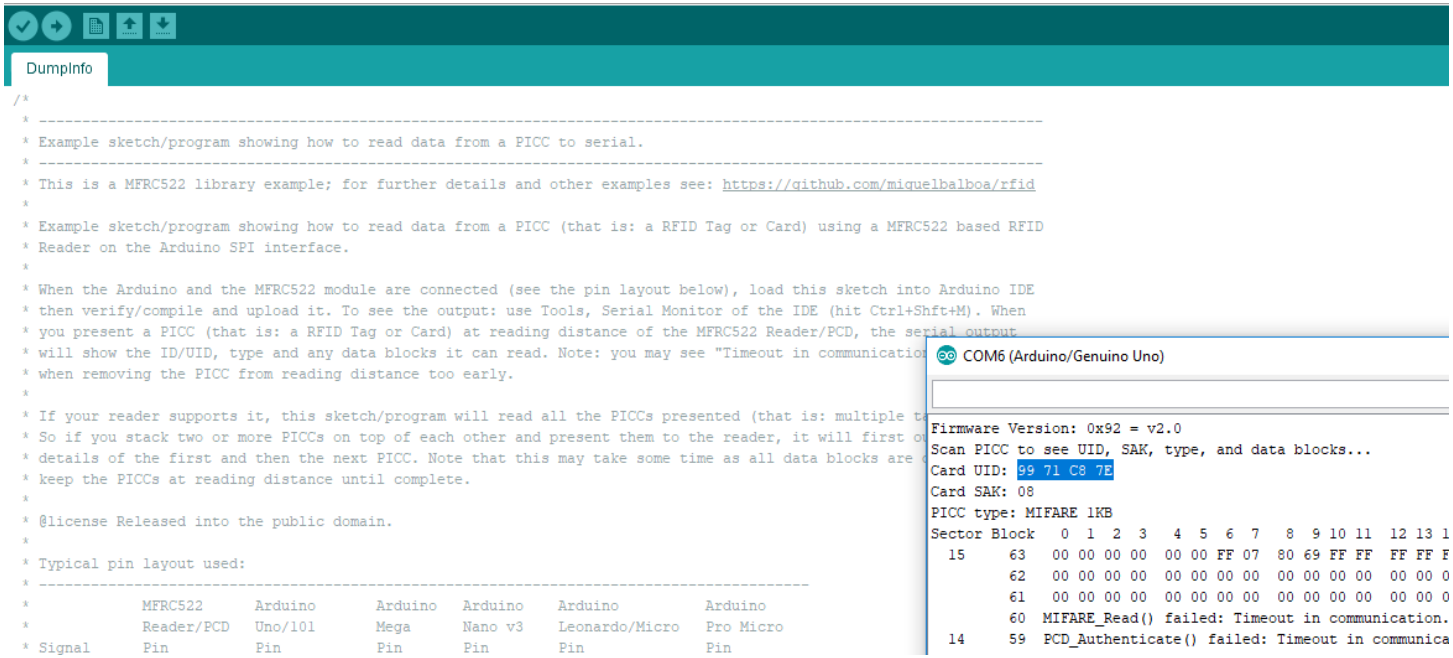
C:\Program Files (x86)\Arduino\libraries\rfid-master\examples\rfid_read_personal_data



# Copy data from Another RFID Tag, Cloner

- **1) Getting data (UID)**

C:\Program Files (x86)\Arduino\libraries\rfid-master\examples\DumpInfo



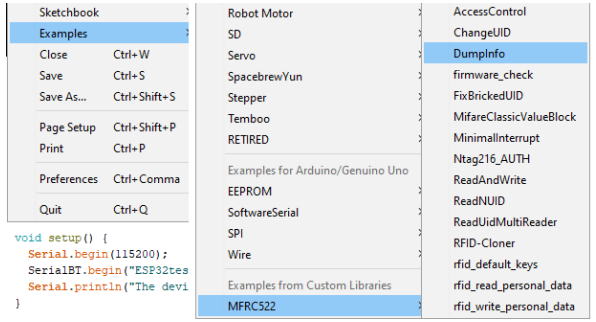
# Copy data from Another RFID Tag, Cloner

- **2) change UID**

C:\Program Files (x86)\Arduino\libraries\rfid-master\examples\ChangeUID

ChangeUID

```
 * SPI MISO     MISO          12 / ICSP-1    50         D12         ICSP-1          14
 * SPI SCK      SCK           13 / ICSP-3    52         D13         ICSP-3          15
 */

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN    9     // Configurable, see typical pin layout above
#define SS_PIN     10    // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance

/* Set your new UID here! */
//#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
#define NEW_UID {0x99, 0x71, 0xC8, 0x7E}  //99 71 C8 7E

MFRC522::MIFARE_Key key;
```




C:\Program Files (x86)\Arduino\libraries\rfid-master\examples\RFID-Cloner

```
Card did not respond to 0x40 after HALT command. Are you sure it is a UID changeable one?
```

# RFID Exercise

- **Multiple tags**
- **RFID ring - https://www.youtube.com/watch?v=_Sj17Lb38e0**
- https://www.adafruit.com/product/2800?utm_source=youtube&utm_medium=videodescrip&utm_campaign=3dprinting

```
if(rfid.uid.uidByte[0] == 0x2A && rfid.uid.uidByte[1] == 0x22 &&
rfid.uid.uidByte[2] == 0xC1 && rfid.uid.uidByte[3] == 0x23)
```