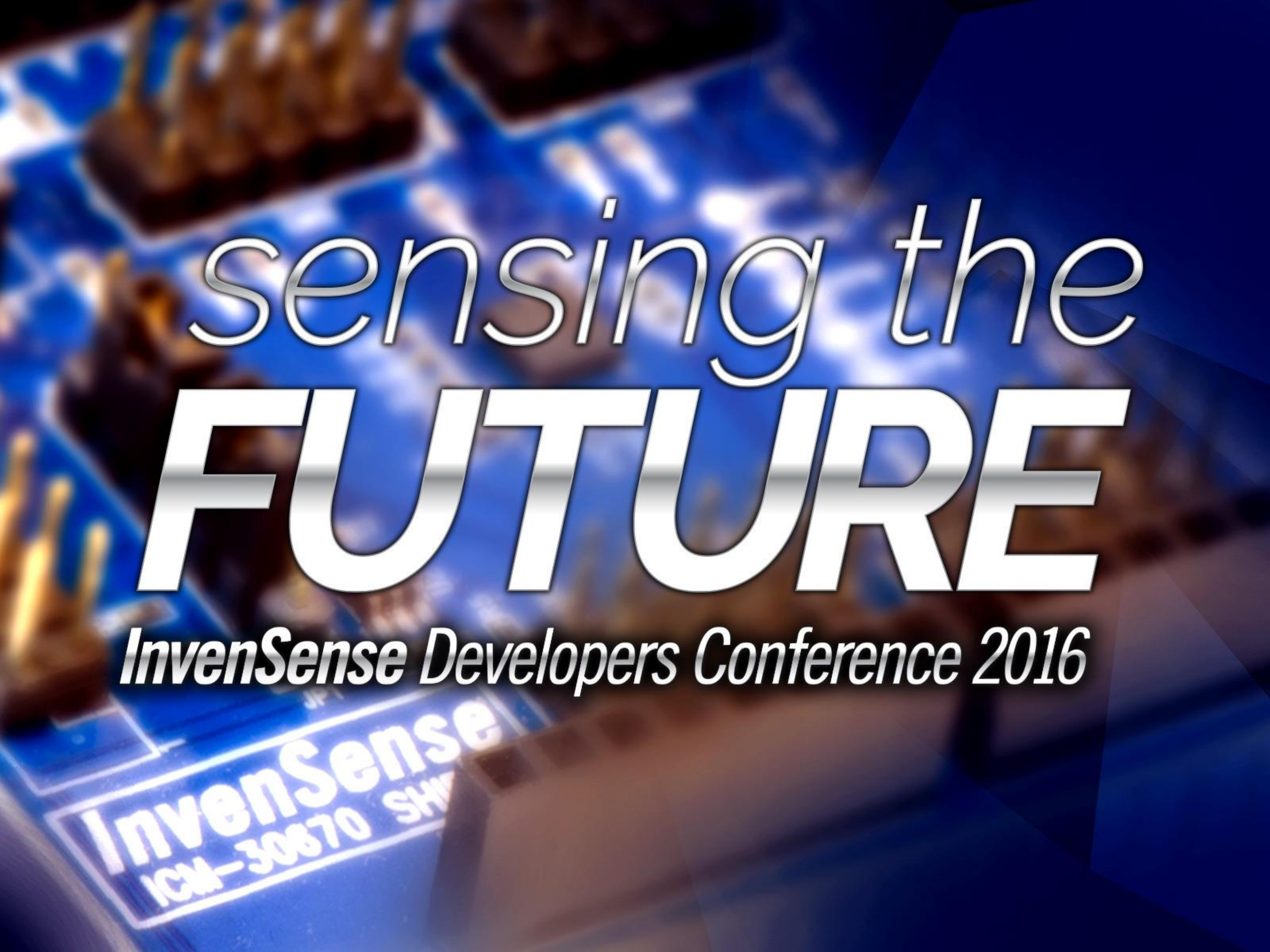
sensing the FUTURE

InvenSense Developers Conference 2016



SensorStudio

With ICM-30670

Joke of the day ☺

- Wondering how we created a PingPong demo?

LET'S GET A PING-PONG TABLE SO WE LOOK LIKE A GREAT PLACE TO WORK.
PUT IT IN A CENTRAL AREA THAT WILL DISRUPT THE ENTIRE FLOOR IF ANYONE USES IT.
I JUST REALIZED THAT I DON'T KNOW WHY NOISE COMES OUT OF MY MOUTH.
reactor.cc

(Source http://joyreactor.com/post/742846)

Foreword

- “And we’ve ***analyzed over 700 swimmers***, different body types, different abilities. We hooked them up to state-of-the-art metabolic equipment. We’ve even drawn blood samples to look at lactic acid levels and we used all this body of information to **create an algorithm** that will give you the most accurate calorie burn information while you’re swimming. ”

Source http://www.singjupost.com/apple-iphone-7-keynote-september-2016-launch-event-full-transcript/4/

Source http://www.apple.com/newsroom/2016/09/apple-introduces-apple-watch-series-2.html

Agenda

- Why
- What
- How: Hardware
- How: Software
- Demo

Why

- Bring a “WOW factor” @ IDC’2016
- Inspire you to create great sport applications

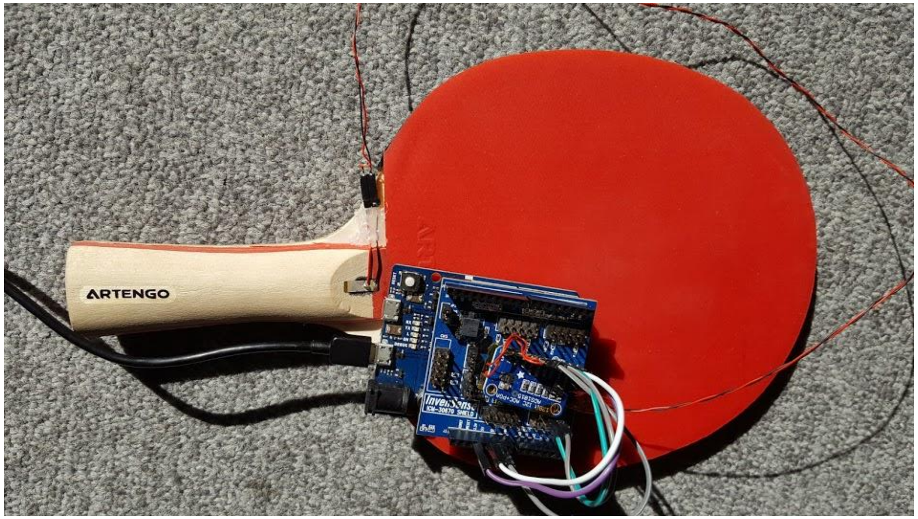
Using SensorStudio & ICM-30670 Dev Kit

What

- Piezzo+ADC
  - Raw signals (used for ball impact)

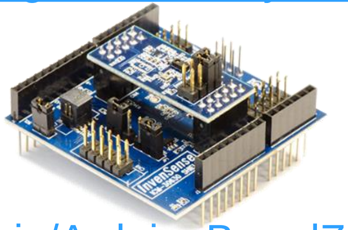
- FireFly ICM-30670
  - Fusion Piezzo & IMU
  - Ball impact detection

- SensorStudio
  - Design/Debug/Demo



How: Hardware BOM

- SensorStudio ICM-30670 Dev Kit - https://www.invensense.com/products/motion-tracking/6-axis/firefly-development-kit/

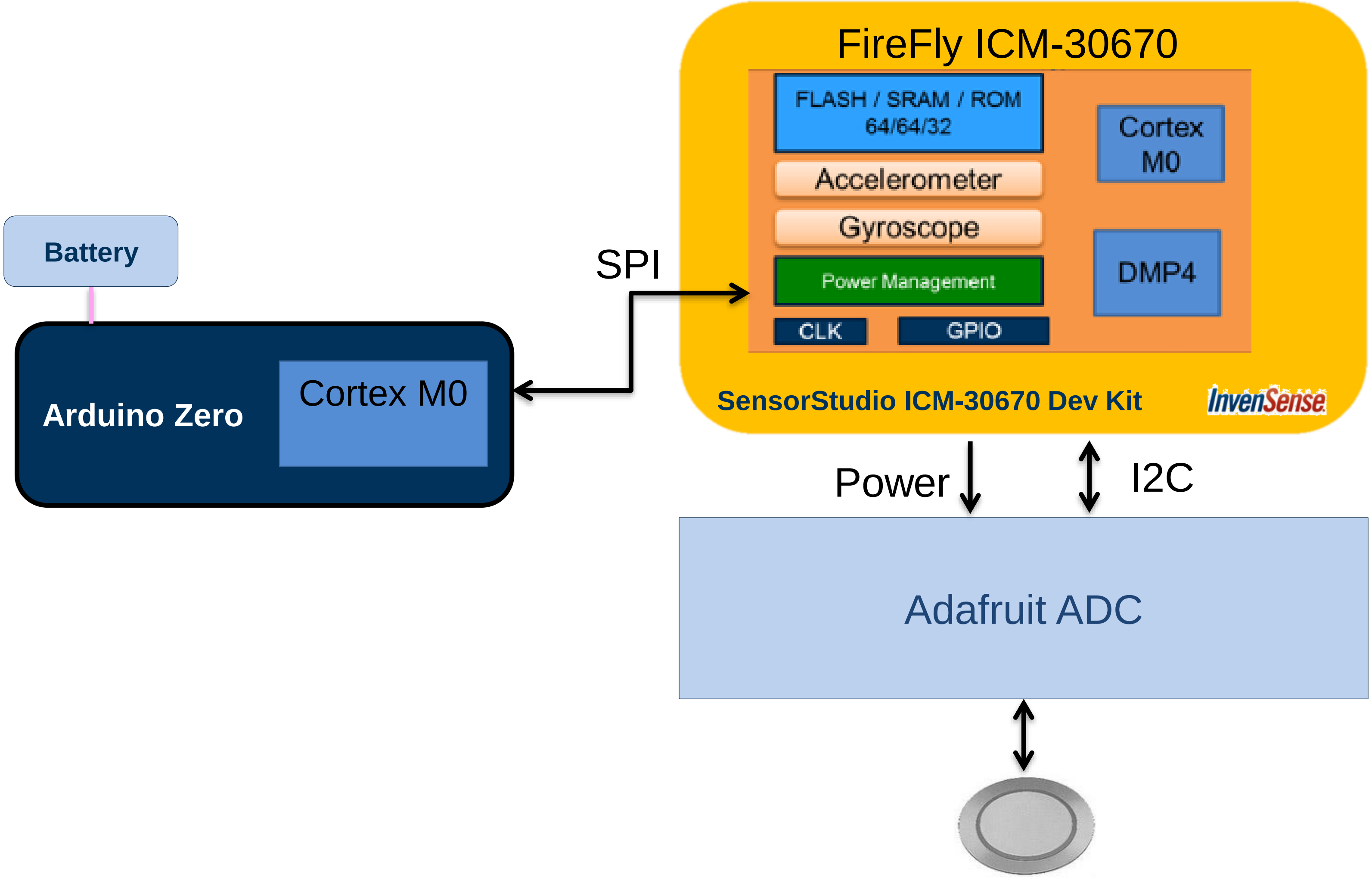


- Arduino Zero - https://www.arduino.cc/en/Main/ArduinoBoardZero

- Adafruit ADC - https://www.adafruit.com/products/1083

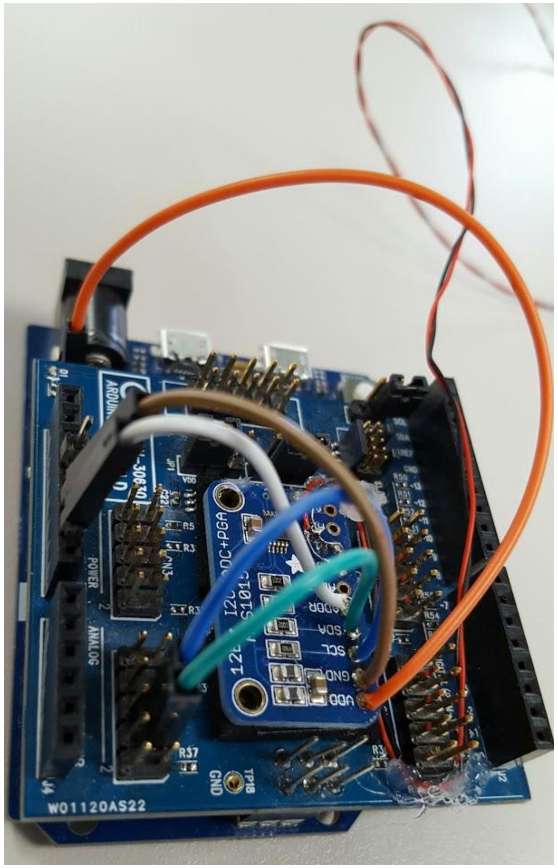
- Piezzo - https://www.arrow.com/en/products/7bb-12-9/murata-manufacturing

How: Hardware Schematic



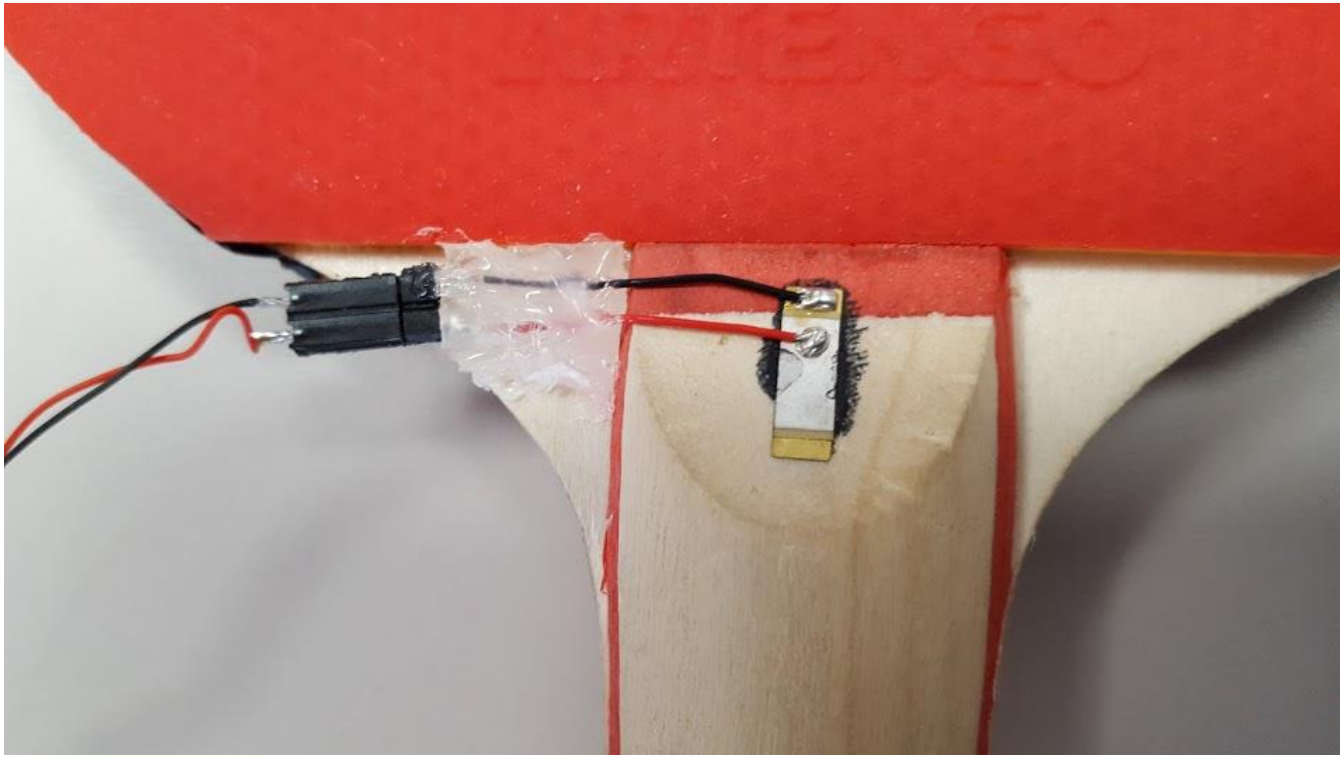
How: Hardware DevKit adaptations

- Connect ICM-30670 DevKit to ADC
  - Power, I2C
- Use double sided foam strip with adhesives on both sides



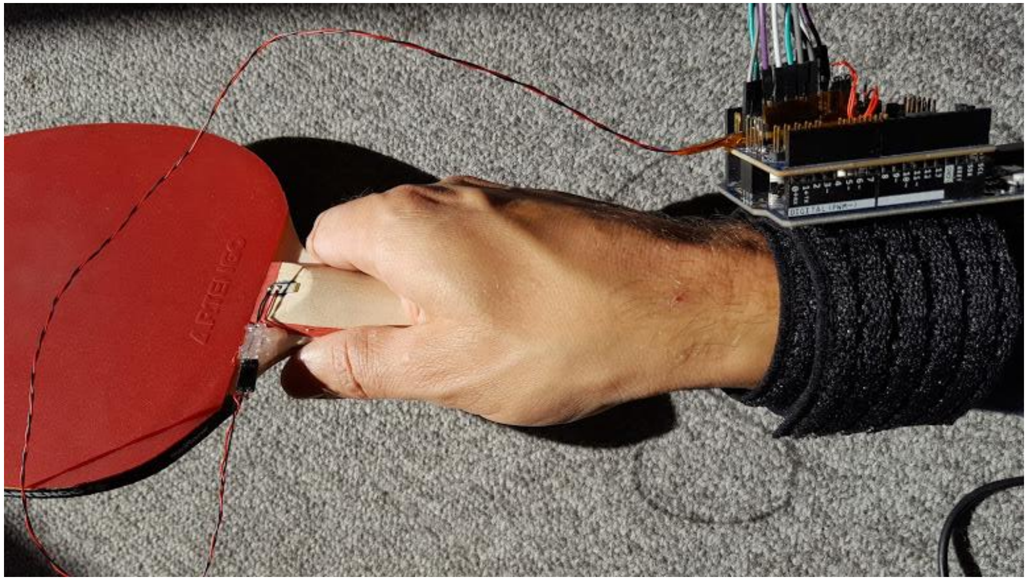
How: Hardware Racket adaptations

- Cut & glue the Piezzo buzzer on the racket
  - Conserve ability to respond to applied mechanical stress
- Connect ADC to Piezzo
  - Make it easy to plug in/out



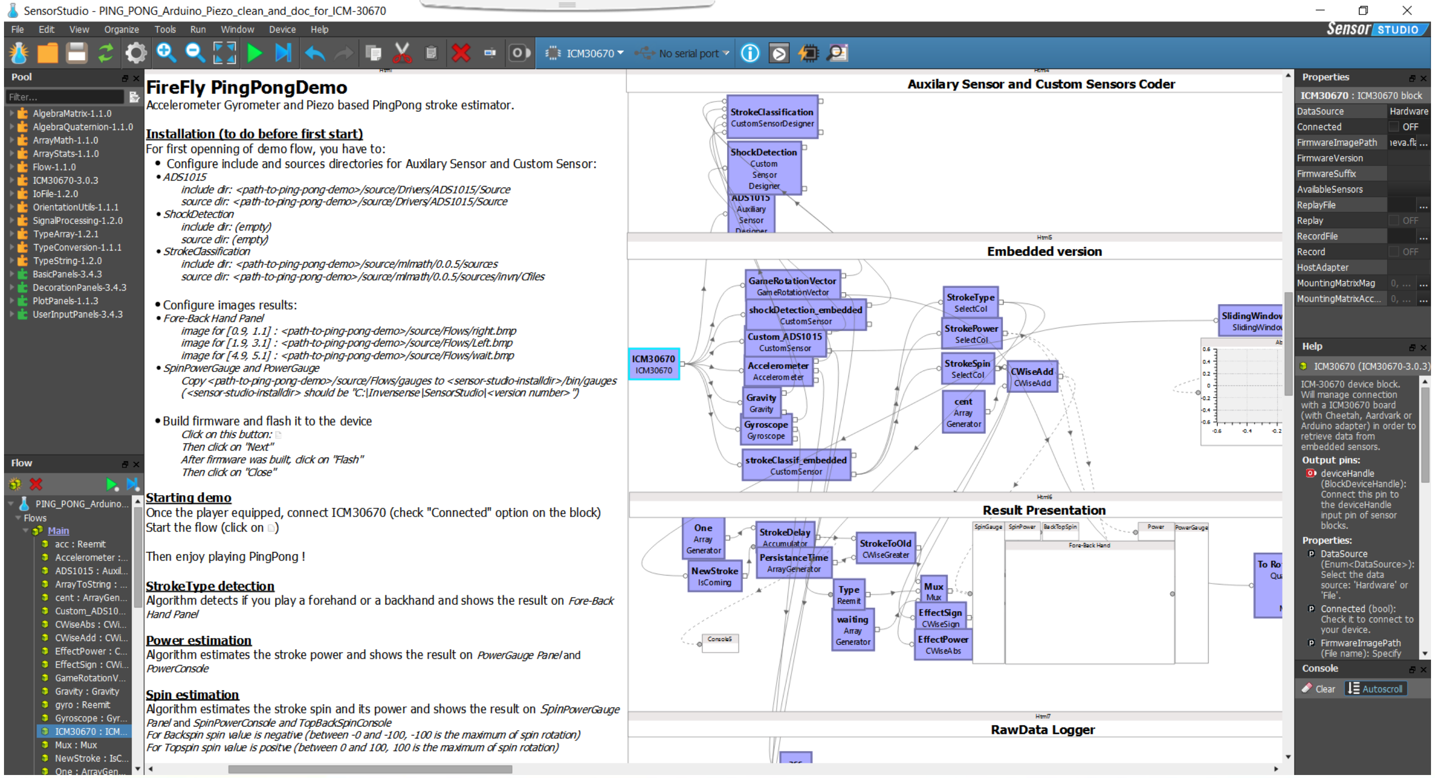
How: Humanoid hack ☺

- Got to pay the price, so science can advance!
  - Elastic band strap with velcro is your friend
- Connect
  - Piezzo to ADC
  - Arduino zero to PC (tie the cable to your body)



How: Software SensorStudio

- SensorStudio used to create Piezzo/ADC driver (AuxiliarySensor)
- SensorStudio used to create algorithm (CustomSensor)
- Visualization of sensors & algorithm outputs



FireFly PingPongDemo

Accelerometer Gyrometer and Piezo based PingPong stroke estimator.

Installation (to do before first start)

For first openning of demo flow, you have to:

- Configure include and sources directories for Auxilary Sensor and Custom Sensor:
- *ADS1015*
  include dir: <path-to-ping-pong-demo>/source/Drivers/ADS1015/Source
  source dir: <path-to-ping-pong-demo>/source/Drivers/ADS1015/Source
- *ShockDetection*
  include dir: (empty)
  source dir: (empty)
- *StrokeClassification*
  include dir: <path-to-ping-pong-demo>/source/mlmath/0.0.5/sources
  source dir: <path-to-ping-pong-demo>/source/mlmath/0.0.5/sources/invn/Cfiles

- Configure images results:
- *Fore-Back Hand Panel*
  image for [0.9, 1.1] : <path-to-ping-pong-demo>/source/Flows/right.bmp
  image for [1.9, 3.1] : <path-to-ping-pong-demo>/source/Flows/Left.bmp
  image for [4.9, 5.1] : <path-to-ping-pong-demo>/source/Flows/wait.bmp
- *SpinPowerGauge and PowerGauge*
  Copy <path-to-ping-pong-demo>/source/Flows/gauges to <sensor-studio-installdir>/bin/gauges (<sensor-studio-installdir> should be "C:|Invensense|SensorStudio|<version number>")

- Build firmware and flash it to the device
  Click on this button:
  Then click on "Next"
  After firmware was built, click on "Flash"
  Then click on "Close"

Starting demo

Once the player equipped, connect ICM30670 (check "Connected" option on the block)
Start the flow (click on)

Then enjoy playing PingPong !

StrokeType detection

Algorithm detects if you play a forehand or a backhand and shows the result on *Fore-Back Hand Panel*

Power estimation

Algorithm estimates the stroke power and shows the result on *PowerGauge Panel* and *PowerConsole*

Spin estimation

Algorithm estimates the stroke spin and its power and shows the result on *SpinPowerGauge Panel* and *SpinPowerConsole and TopBackSpinConsole*
For Backspin spin value is negative (between -0 and -100, -100 is the maximum of spin rotation)
For Topspin spin value is positve (between 0 and 100, 100 is the maximum of spin rotation)

ICM30670 : ICM30670 block

DataSource	Hardware
Connected	□ OFF
FirmwareImagePath	ıeva.fl ...
FirmwareVersion	
FirmwareSuffix	
AvailableSensors	
ReplayFile	...
Replay	□ OFF
RecordFile	...
Record	□ OFF
HostAdapter	
MountingMatrixMag	0, ...
MountingMatrixAcc...	0, ...

ICM30670 (ICM30670-3.0.3)

ICM-30670 device block. Will manage connection with a ICM30670 board (with Cheetah, Aardvark or Arduino adapter) in order to retrieve data from embedded sensors.

Output pins:

deviceHandle (BlockDeviceHandle): Connect this pin to the deviceHandle input pin of sensor blocks.

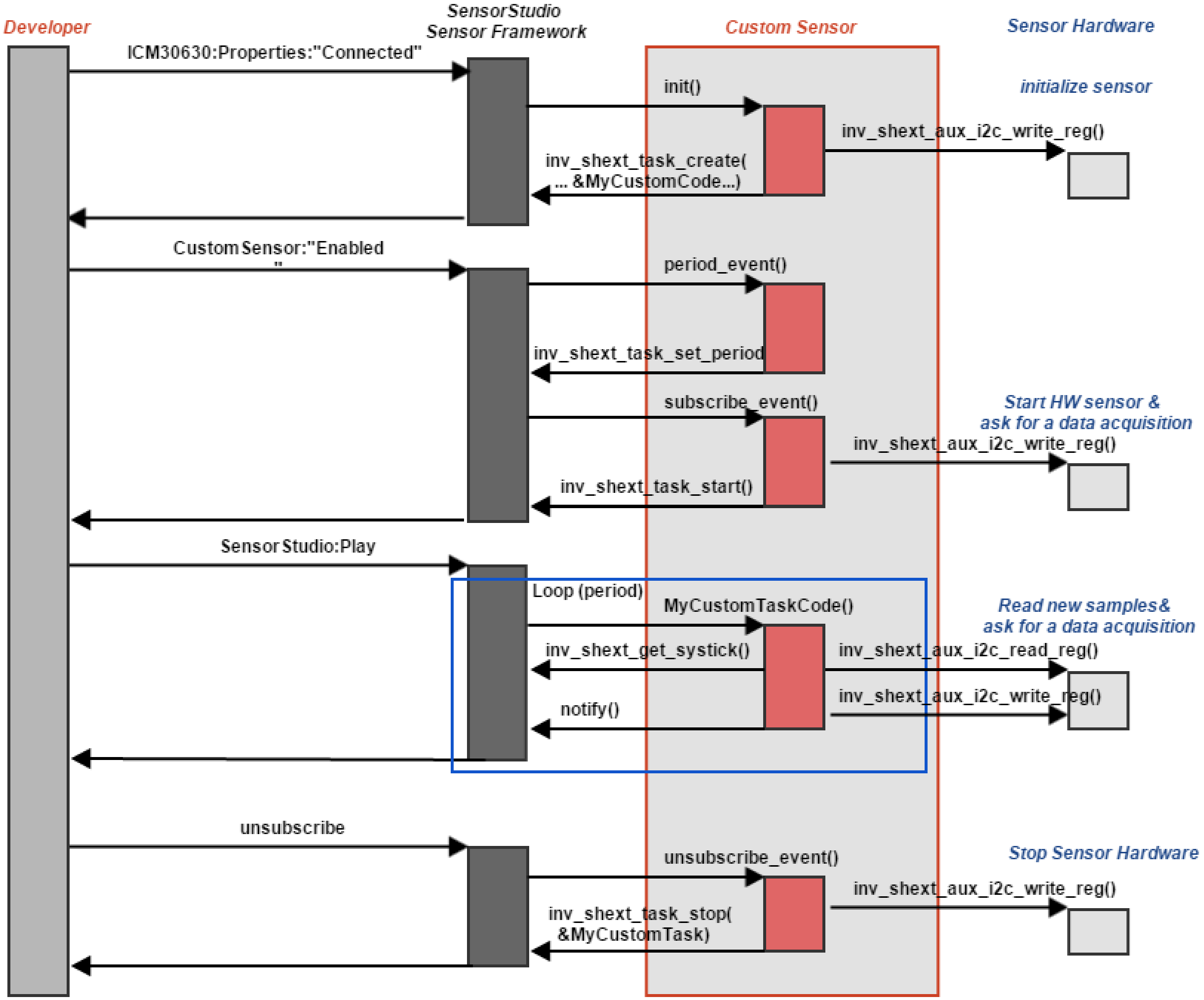
Properties:

- DataSource (Enum<DataSource>): Select the data source: 'Hardware' or 'File'.
- Connected (bool): Check it to connect to your device.
- FirmwareImagePath (File name): Specify

Reference: Add a driver

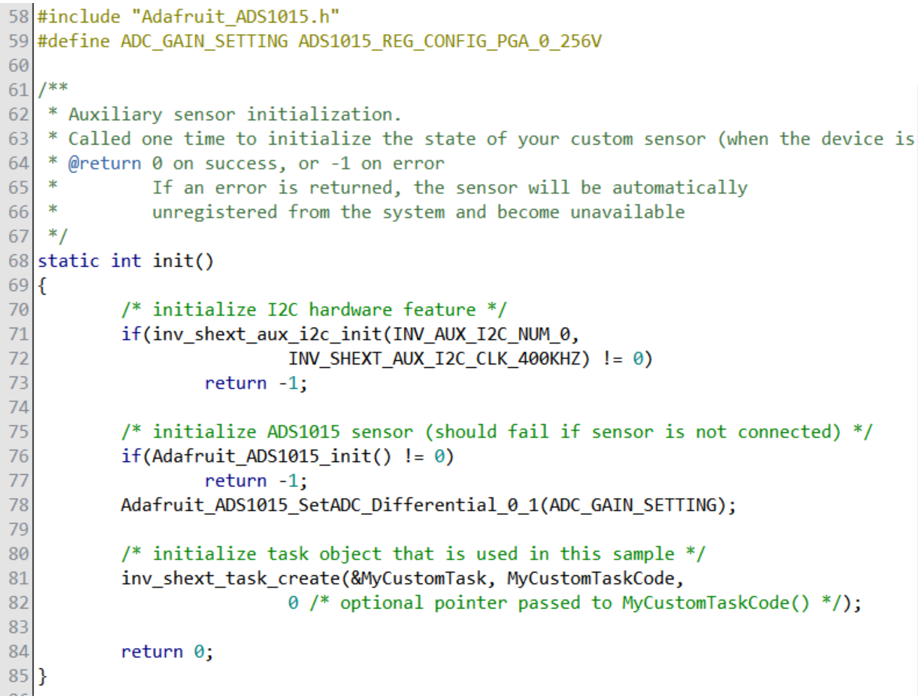
sensing the FUTURE

- **Task & notify** pattern to produce sensor data

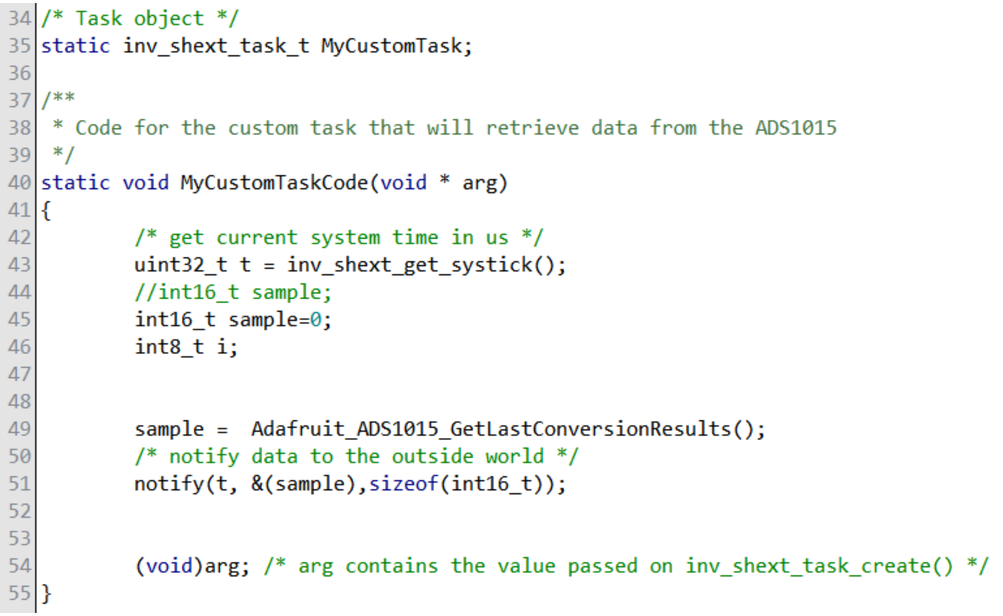

Developer
SensorStudio Sensor Framework
Custom Sensor
Sensor Hardware
ICM30630:Properties:"Connected"
init()
initialize sensor
inv_shext_aux_i2c_write_reg()
inv_shext_task_create(... &MyCustomCode...)
CustomSensor:"Enabled"
period_event()
inv_shext_task_set_period
subscribe_event()
Start HW sensor & ask for a data acquisition
inv_shext_aux_i2c_write_reg()
inv_shext_task_start()
SensorStudio:Play
Loop (period)
MyCustomTaskCode()
Read new samples& ask for a data acquisition
inv_shext_get_systick()
inv_shext_aux_i2c_read_reg()
inv_shext_aux_i2c_write_reg()
notify()
unsubscribe
unsubscribe_event()
Stop Sensor Hardware
inv_shext_aux_i2c_write_reg()
inv_shext_task_stop(&MyCustomTask)

How: Software Piezzo/ADC driver

- Configure the I2C
- Initialize ADC, set its range to 256mV
- Starts the acquisition task
  - notify sensor hub of the new piezo data



```c
#include "Adafruit_ADS1015.h"
#define ADC_GAIN_SETTING ADS1015_REG_CONFIG_PGA_0_256V

/**
 * Auxiliary sensor initialization.
 * Called one time to initialize the state of your custom sensor (when the device is
 * @return 0 on success, or -1 on error
 *         If an error is returned, the sensor will be automatically
 *         unregistered from the system and become unavailable
 */
static int init()
{
        /* initialize I2C hardware feature */
        if(inv_shext_aux_i2c_init(INV_AUX_I2C_NUM_0,
                        INV_SHEXT_AUX_I2C_CLK_400KHZ) != 0)
                return -1;

        /* initialize ADS1015 sensor (should fail if sensor is not connected) */
        if(Adafruit_ADS1015_init() != 0)
                return -1;
        Adafruit_ADS1015_SetADC_Differential_0_1(ADC_GAIN_SETTING);

        /* initialize task object that is used in this sample */
        inv_shext_task_create(&MyCustomTask, MyCustomTaskCode,
                        0 /* optional pointer passed to MyCustomTaskCode() */);

        return 0;
}
```

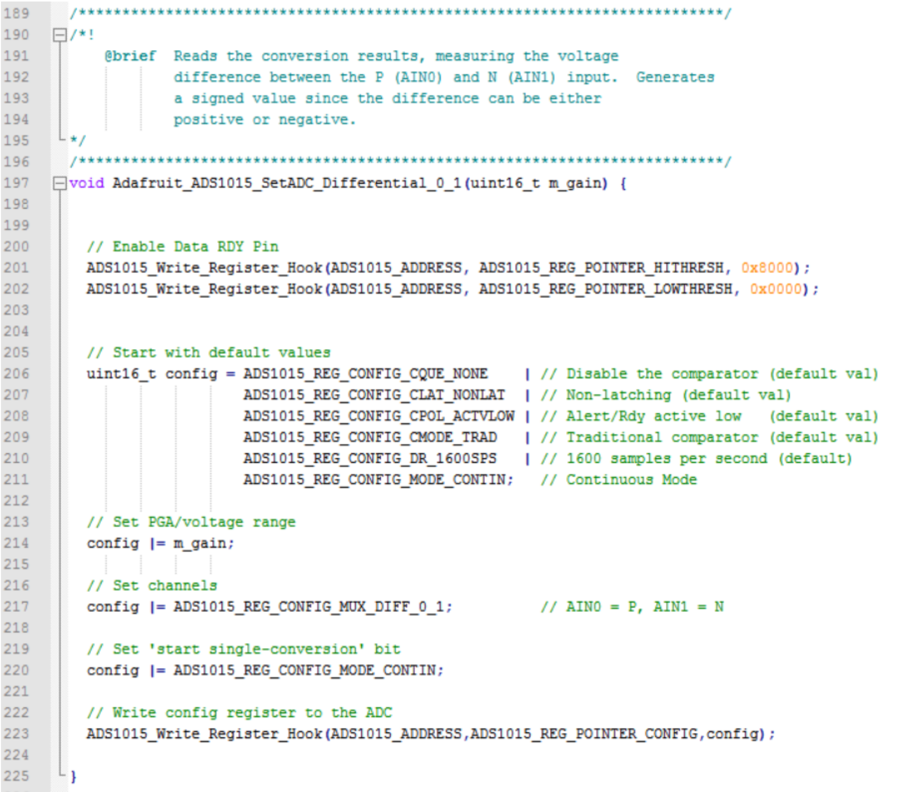


```c
/* Task object */
static inv_shext_task_t MyCustomTask;

/**
 * Code for the custom task that will retrieve data from the ADS1015
 */
static void MyCustomTaskCode(void * arg)
{
        /* get current system time in us */
        uint32_t t = inv_shext_get_systick();
        //int16_t sample;
        int16_t sample=0;
        int8_t i;


        sample =  Adafruit_ADS1015_GetLastConversionResults();
        /* notify data to the outside world */
        notify(t, &(sample),sizeof(int16_t));


        (void)arg; /* arg contains the value passed on inv_shext_task_create() */
}
```

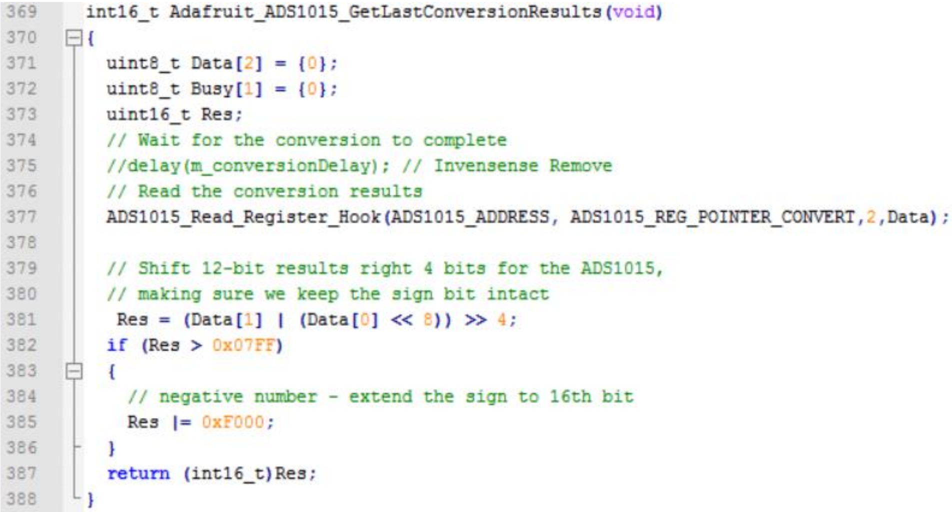
How: Software low level driver

sensing the FUTURE

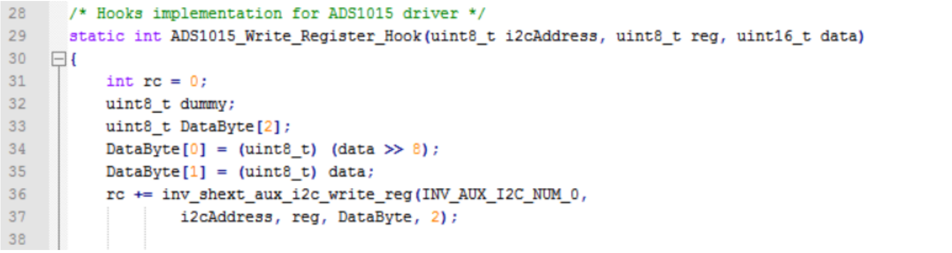
- All come down to read/write on I2C



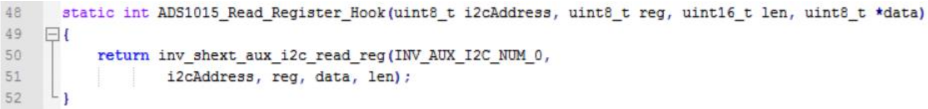
```
/**************************************************************************/
/*!
    @brief  Reads the conversion results, measuring the voltage
            difference between the P (AIN0) and N (AIN1) input.  Generates
            a signed value since the difference can be either
            positive or negative.
*/
/**************************************************************************/
void Adafruit_ADS1015_SetADC_Differential_0_1(uint16_t m_gain) {


  // Enable Data RDY Pin
  ADS1015_Write_Register_Hook(ADS1015_ADDRESS, ADS1015_REG_POINTER_HITHRESH, 0x8000);
  ADS1015_Write_Register_Hook(ADS1015_ADDRESS, ADS1015_REG_POINTER_LOWTHRESH, 0x0000);


  // Start with default values
  uint16_t config = ADS1015_REG_CONFIG_CQUE_NONE    | // Disable the comparator (default val)
                    ADS1015_REG_CONFIG_CLAT_NONLAT  | // Non-latching (default val)
                    ADS1015_REG_CONFIG_CPOL_ACTVLOW | // Alert/Rdy active low   (default val)
                    ADS1015_REG_CONFIG_CMODE_TRAD   | // Traditional comparator (default val)
                    ADS1015_REG_CONFIG_DR_1600SPS   | // 1600 samples per second (default)
                    ADS1015_REG_CONFIG_MODE_CONTIN;   // Continuous Mode

  // Set PGA/voltage range
  config |= m_gain;

  // Set channels
  config |= ADS1015_REG_CONFIG_MUX_DIFF_0_1;          // AIN0 = P, AIN1 = N

  // Set 'start single-conversion' bit
  config |= ADS1015_REG_CONFIG_MODE_CONTIN;

  // Write config register to the ADC
  ADS1015_Write_Register_Hook(ADS1015_ADDRESS,ADS1015_REG_POINTER_CONFIG,config);

}
```



```
int16_t Adafruit_ADS1015_GetLastConversionResults(void)
{
  uint8_t Data[2] = {0};
  uint8_t Busy[1] = {0};
  uint16_t Res;
  // Wait for the conversion to complete
  //delay(m_conversionDelay); // Invensense Remove
  // Read the conversion results
  ADS1015_Read_Register_Hook(ADS1015_ADDRESS, ADS1015_REG_POINTER_CONVERT,2,Data);

  // Shift 12-bit results right 4 bits for the ADS1015,
  // making sure we keep the sign bit intact
   Res = (Data[1] | (Data[0] << 8)) >> 4;
  if (Res > 0x07FF)
  {
    // negative number - extend the sign to 16th bit
    Res |= 0xF000;
  }
  return (int16_t)Res;
}
```



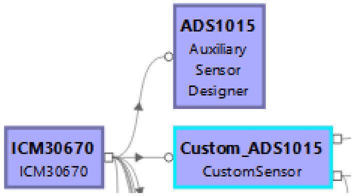
```
/* Hooks implementation for ADS1015 driver */
static int ADS1015_Write_Register_Hook(uint8_t i2cAddress, uint8_t reg, uint16_t data)
{
    int rc = 0;
    uint8_t dummy;
    uint8_t DataByte[2];
    DataByte[0] = (uint8_t) (data >> 8);
    DataByte[1] = (uint8_t) data;
    rc += inv_shext_aux_i2c_write_reg(INV_AUX_I2C_NUM_0,
            i2cAddress, reg, DataByte, 2);

```

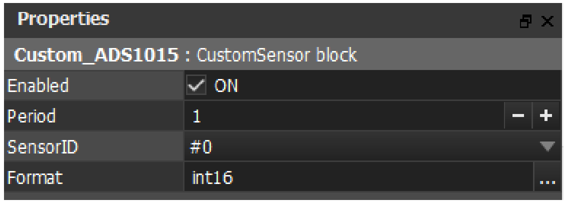


```
static int ADS1015_Read_Register_Hook(uint8_t i2cAddress, uint8_t reg, uint16_t len, uint8_t *data)
{
    return inv_shext_aux_i2c_read_reg(INV_AUX_I2C_NUM_0,
            i2cAddress, reg, data, len);
}
```

How: Software desktop run

- Ball impact detection algorithm need ADC/Piezzo driver

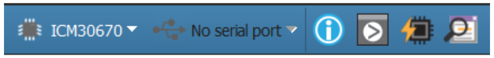


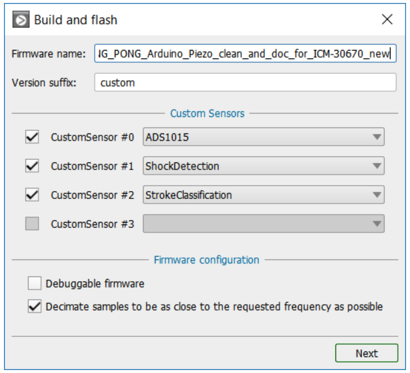

Properties

Custom_ADS1015 : CustomSensor block

Enabled	☑ ON
Period	1
SensorID	#0
Format	int16

- Build & Flash




Build and flash

Firmware name: IG_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new

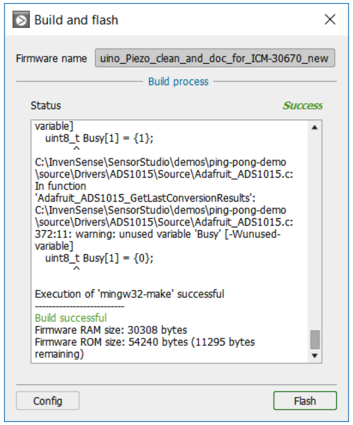
Version suffix: custom

Custom Sensors

☑ CustomSensor #0 ADS1015
☑ CustomSensor #1 ShockDetection
☑ CustomSensor #2 StrokeClassification
□ CustomSensor #3

Firmware configuration

□ Debuggable firmware
☑ Decimate samples to be as close to the requested frequency as possible

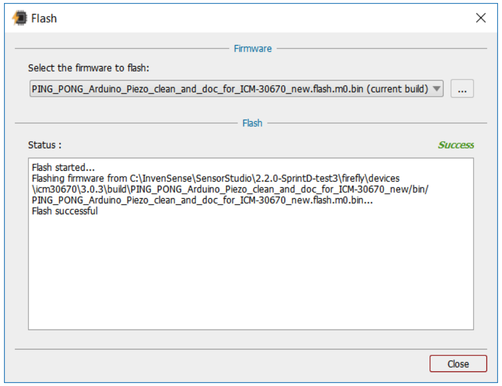

Build and flash

Firmware name uino_Piezo_clean_and_doc_for_ICM-30670_new

Build process

Status Success

```
variable]
   uint8_t Busy[1] = {1};
          ^
C:\InvenSense\SensorStudio\demos\ping-pong-demo
\source\Drivers\ADS1015\Source\Adafruit_ADS1015.c:
In function
'Adafruit_ADS1015_GetLastConversionResults':
C:\InvenSense\SensorStudio\demos\ping-pong-demo
\source\Drivers\ADS1015\Source\Adafruit_ADS1015.c:
372:11: warning: unused variable 'Busy' [-Wunused-
variable]
   uint8_t Busy[1] = {0};
          ^


Execution of 'mingw32-make' successful
--------------------------
Build successful
Firmware RAM size: 30308 bytes
Firmware ROM size: 54240 bytes (11295 bytes
remaining)
```


Flash

Firmware

Select the firmware to flash:

PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new.flash.m0.bin (current build)

Flash

Status : Success

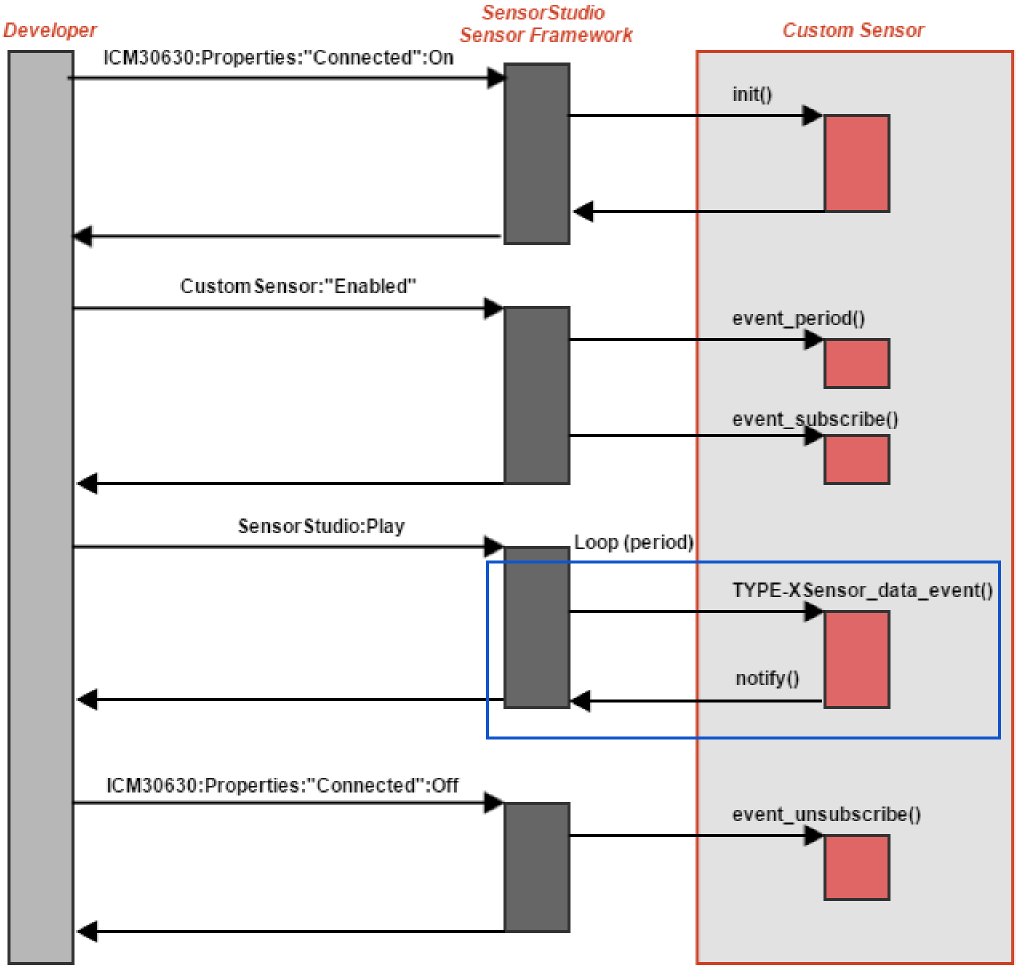
```
Flash started...
Flashing firmware from C:\InvenSense\SensorStudio\2.2.0-SprintD-test3\firefly\devices
\icm30670\3.0.3\build\PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new/bin/
PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new.flash.m0.bin...
Flash successful
```

- Outputs: int16 (Shot)

Reference: Add an algorithm

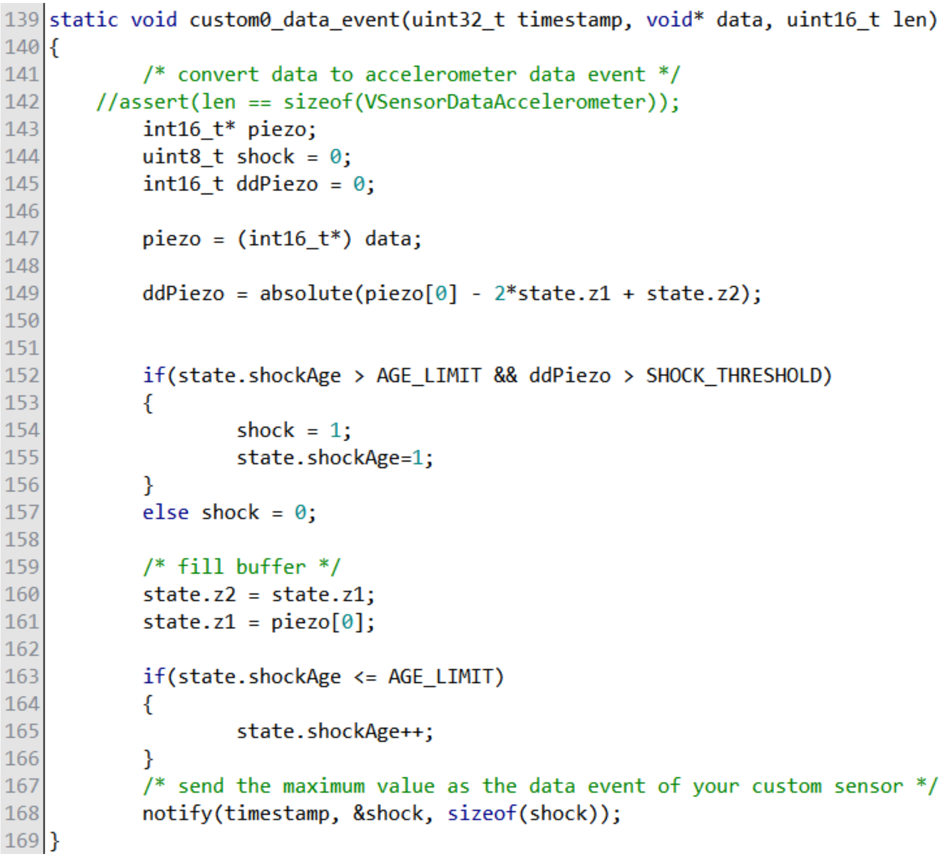
sensing the FUTURE

- **notify** & **subscribe** pattern to consume/produce sensor data



How: Software ball impact detection

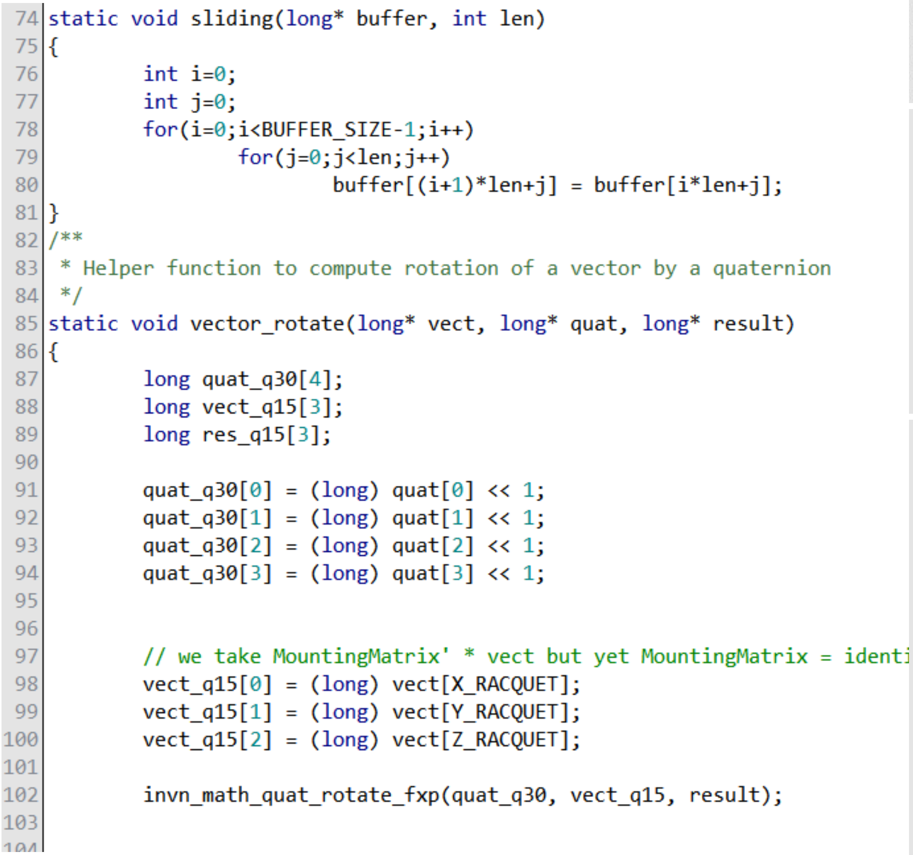
- Principle: Simple Piezzo threshold over time



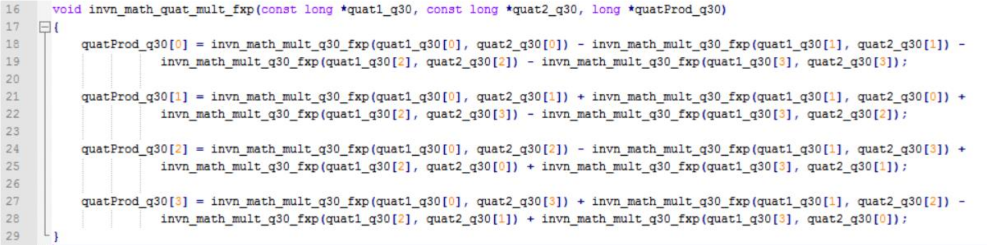
```c
static void custom0_data_event(uint32_t timestamp, void* data, uint16_t len)
{
        /* convert data to accelerometer data event */
    //assert(len == sizeof(VSensorDataAccelerometer));
        int16_t* piezo;
        uint8_t shock = 0;
        int16_t ddPiezo = 0;

        piezo = (int16_t*) data;

        ddPiezo = absolute(piezo[0] - 2*state.z1 + state.z2);


        if(state.shockAge > AGE_LIMIT && ddPiezo > SHOCK_THRESHOLD)
        {
                shock = 1;
                state.shockAge=1;
        }
        else shock = 0;

        /* fill buffer */
        state.z2 = state.z1;
        state.z1 = piezo[0];

        if(state.shockAge <= AGE_LIMIT)
        {
                state.shockAge++;
        }
        /* send the maximum value as the data event of your custom sensor */
        notify(timestamp, &shock, sizeof(shock));
}
```

How: Software helper

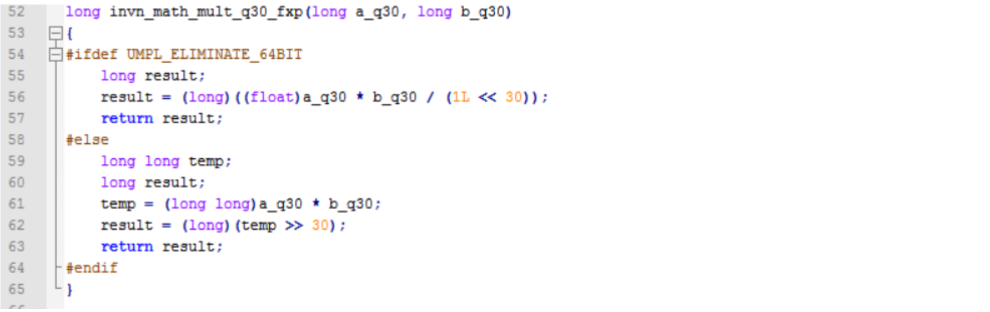
- Principle: Rotate gyro to hearth reference frame



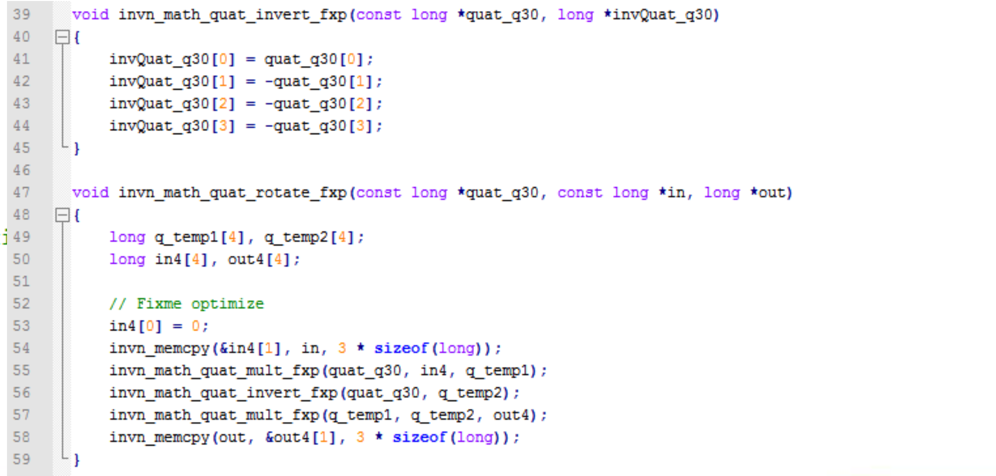
```
static void sliding(long* buffer, int len)
{
        int i=0;
        int j=0;
        for(i=0;i<BUFFER_SIZE-1;i++)
                for(j=0;j<len;j++)
                        buffer[(i+1)*len+j] = buffer[i*len+j];
}
/**
 * Helper function to compute rotation of a vector by a quaternion
 */
static void vector_rotate(long* vect, long* quat, long* result)
{
        long quat_q30[4];
        long vect_q15[3];
        long res_q15[3];

        quat_q30[0] = (long) quat[0] << 1;
        quat_q30[1] = (long) quat[1] << 1;
        quat_q30[2] = (long) quat[2] << 1;
        quat_q30[3] = (long) quat[3] << 1;


        // we take MountingMatrix' * vect but yet MountingMatrix = identi
        vect_q15[0] = (long) vect[X_RACQUET];
        vect_q15[1] = (long) vect[Y_RACQUET];
        vect_q15[2] = (long) vect[Z_RACQUET];

        invn_math_quat_rotate_fxp(quat_q30, vect_q15, result);

```



```
void invn_math_quat_mult_fxp(const long *quat1_q30, const long *quat2_q30, long *quatProd_q30)
{
    quatProd_q30[0] = invn_math_mult_q30_fxp(quat1_q30[0], quat2_q30[0]) - invn_math_mult_q30_fxp(quat1_q30[1], quat2_q30[1]) -
                invn_math_mult_q30_fxp(quat1_q30[2], quat2_q30[2]) - invn_math_mult_q30_fxp(quat1_q30[3], quat2_q30[3]);

    quatProd_q30[1] = invn_math_mult_q30_fxp(quat1_q30[0], quat2_q30[1]) + invn_math_mult_q30_fxp(quat1_q30[1], quat2_q30[0]) +
                invn_math_mult_q30_fxp(quat1_q30[2], quat2_q30[3]) - invn_math_mult_q30_fxp(quat1_q30[3], quat2_q30[2]);

    quatProd_q30[2] = invn_math_mult_q30_fxp(quat1_q30[0], quat2_q30[2]) - invn_math_mult_q30_fxp(quat1_q30[1], quat2_q30[3]) +
                invn_math_mult_q30_fxp(quat1_q30[2], quat2_q30[0]) + invn_math_mult_q30_fxp(quat1_q30[3], quat2_q30[1]);

    quatProd_q30[3] = invn_math_mult_q30_fxp(quat1_q30[0], quat2_q30[3]) + invn_math_mult_q30_fxp(quat1_q30[1], quat2_q30[2]) -
                invn_math_mult_q30_fxp(quat1_q30[2], quat2_q30[1]) + invn_math_mult_q30_fxp(quat1_q30[3], quat2_q30[0]);
}
```



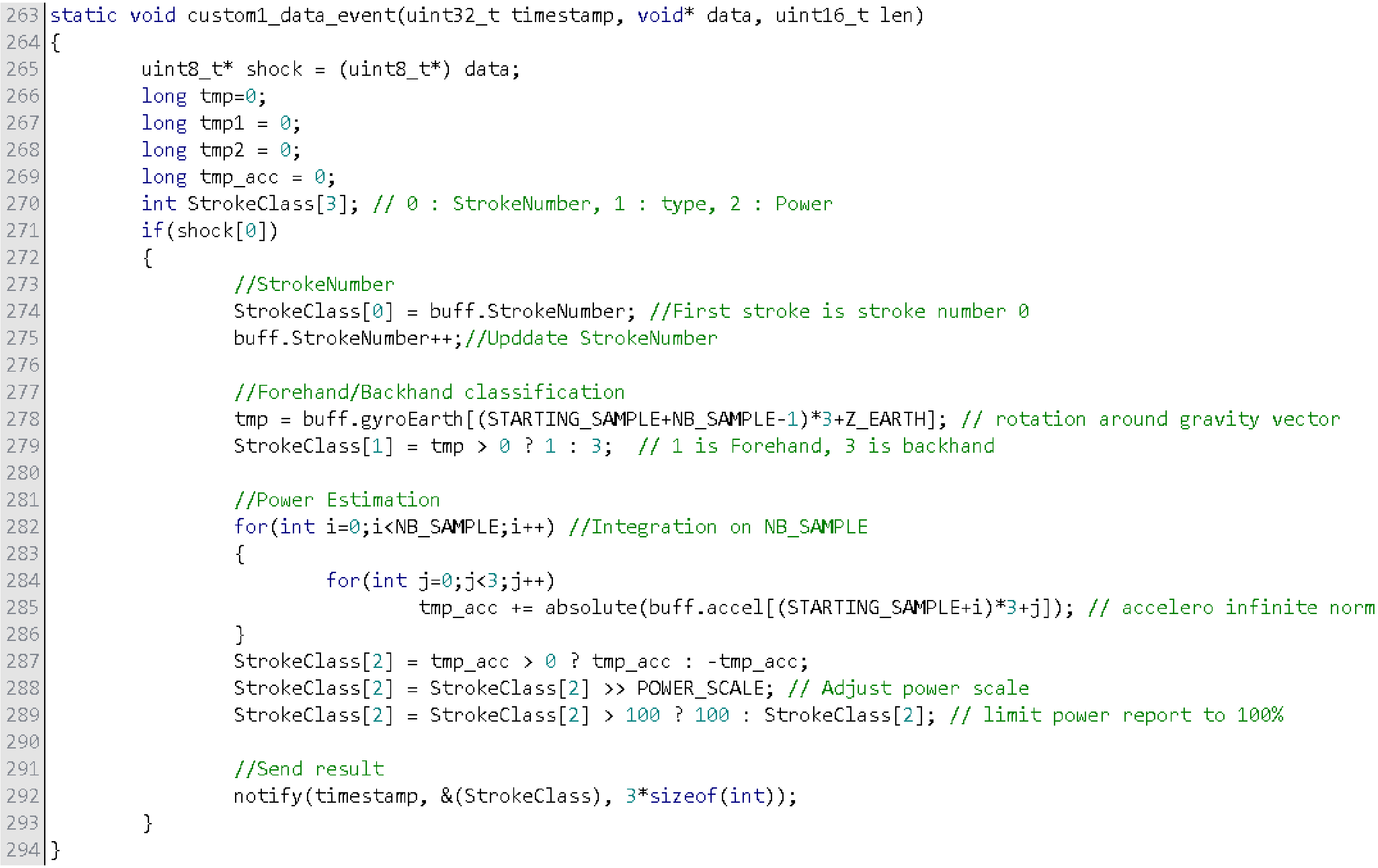
```
long invn_math_mult_q30_fxp(long a_q30, long b_q30)
{
#ifdef UMPL_ELIMINATE_64BIT
    long result;
    result = (long)((float)a_q30 * b_q30 / (1L << 30));
    return result;
#else
    long long temp;
    long result;
    temp = (long long)a_q30 * b_q30;
    result = (long)(temp >> 30);
    return result;
#endif
}
```



```
void invn_math_quat_invert_fxp(const long *quat_q30, long *invQuat_q30)
{
    invQuat_q30[0] = quat_q30[0];
    invQuat_q30[1] = -quat_q30[1];
    invQuat_q30[2] = -quat_q30[2];
    invQuat_q30[3] = -quat_q30[3];
}

void invn_math_quat_rotate_fxp(const long *quat_q30, const long *in, long *out)
{
    long q_temp1[4], q_temp2[4];
    long in4[4], out4[4];

    // Fixme optimize
    in4[0] = 0;
    invn_memcpy(&in4[1], in, 3 * sizeof(long));
    invn_math_quat_mult_fxp(quat_q30, in4, q_temp1);
    invn_math_quat_invert_fxp(quat_q30, q_temp2);
    invn_math_quat_mult_fxp(q_temp1, q_temp2, out4);
    invn_memcpy(out, &out4[1], 3 * sizeof(long));
}
```

How: Software Fore/Back hand detection

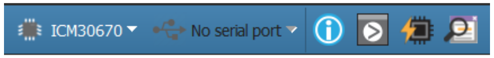
- Principle: Rotation sign around gravity vector

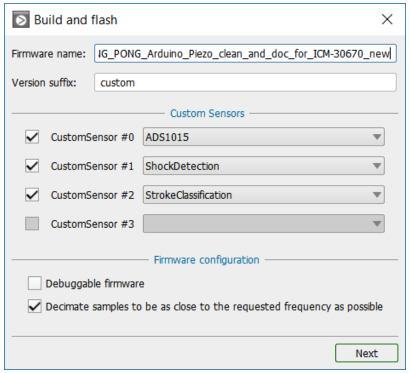


```c
static void custom1_data_event(uint32_t timestamp, void* data, uint16_t len)
{
        uint8_t* shock = (uint8_t*) data;
        long tmp=0;
        long tmp1 = 0;
        long tmp2 = 0;
        long tmp_acc = 0;
        int StrokeClass[3]; // 0 : StrokeNumber, 1 : type, 2 : Power
        if(shock[0])
        {
                //StrokeNumber
                StrokeClass[0] = buff.StrokeNumber; //First stroke is stroke number 0
                buff.StrokeNumber++;//Upddate StrokeNumber

                //Forehand/Backhand classification
                tmp = buff.gyroEarth[(STARTING_SAMPLE+NB_SAMPLE-1)*3+Z_EARTH]; // rotation around gravity vector
                StrokeClass[1] = tmp > 0 ? 1 : 3;  // 1 is Forehand, 3 is backhand

                //Power Estimation
                for(int i=0;i<NB_SAMPLE;i++) //Integration on NB_SAMPLE
                {
                        for(int j=0;j<3;j++)
                                tmp_acc += absolute(buff.accel[(STARTING_SAMPLE+i)*3+j]); // accelero infinite norm
                }
                StrokeClass[2] = tmp_acc > 0 ? tmp_acc : -tmp_acc;
                StrokeClass[2] = StrokeClass[2] >> POWER_SCALE; // Adjust power scale
                StrokeClass[2] = StrokeClass[2] > 100 ? 100 : StrokeClass[2]; // limit power report to 100%

                //Send result
                notify(timestamp, &(StrokeClass), 3*sizeof(int));
        }
}
```

How: Software embedded run

- Build & Flash Shot Classification algorithm





Build and flash

Firmware name: IG_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new

Version suffix: custom

Custom Sensors

☑ CustomSensor #0 ADS1015

☑ CustomSensor #1 ShockDetection

☑ CustomSensor #2 StrokeClassification

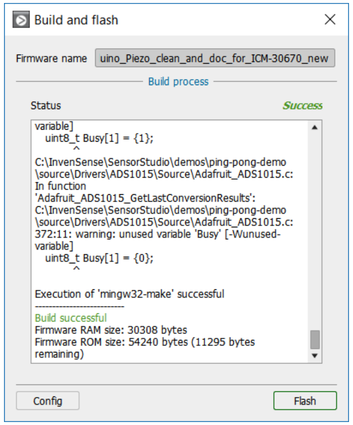
□ CustomSensor #3

Firmware configuration

□ Debuggable firmware

☑ Decimate samples to be as close to the requested frequency as possible

Next



Build and flash

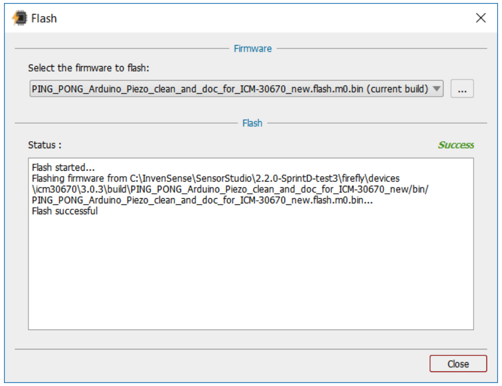
Firmware name uino_Piezo_clean_and_doc_for_ICM-30670_new

Build process

Status Success

```
variable]
   uint8_t Busy[1] = {1};
         ^
C:\InvenSense\SensorStudio\demos\ping-pong-demo
\source\Drivers\ADS1015\Source\Adafruit_ADS1015.c:
In function
'Adafruit_ADS1015_GetLastConversionResults':
C:\InvenSense\SensorStudio\demos\ping-pong-demo
\source\Drivers\ADS1015\Source\Adafruit_ADS1015.c:
372:11: warning: unused variable 'Busy' [-Wunused-
variable]
   uint8_t Busy[1] = {0};
         ^


Execution of 'mingw32-make' successful
-------------------------
Build successful
Firmware RAM size: 30308 bytes
Firmware ROM size: 54240 bytes (11295 bytes
remaining)
```

Config Flash



Flash

Firmware

Select the firmware to flash:

PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new.flash.m0.bin (current build) ...

Flash

Status : Success

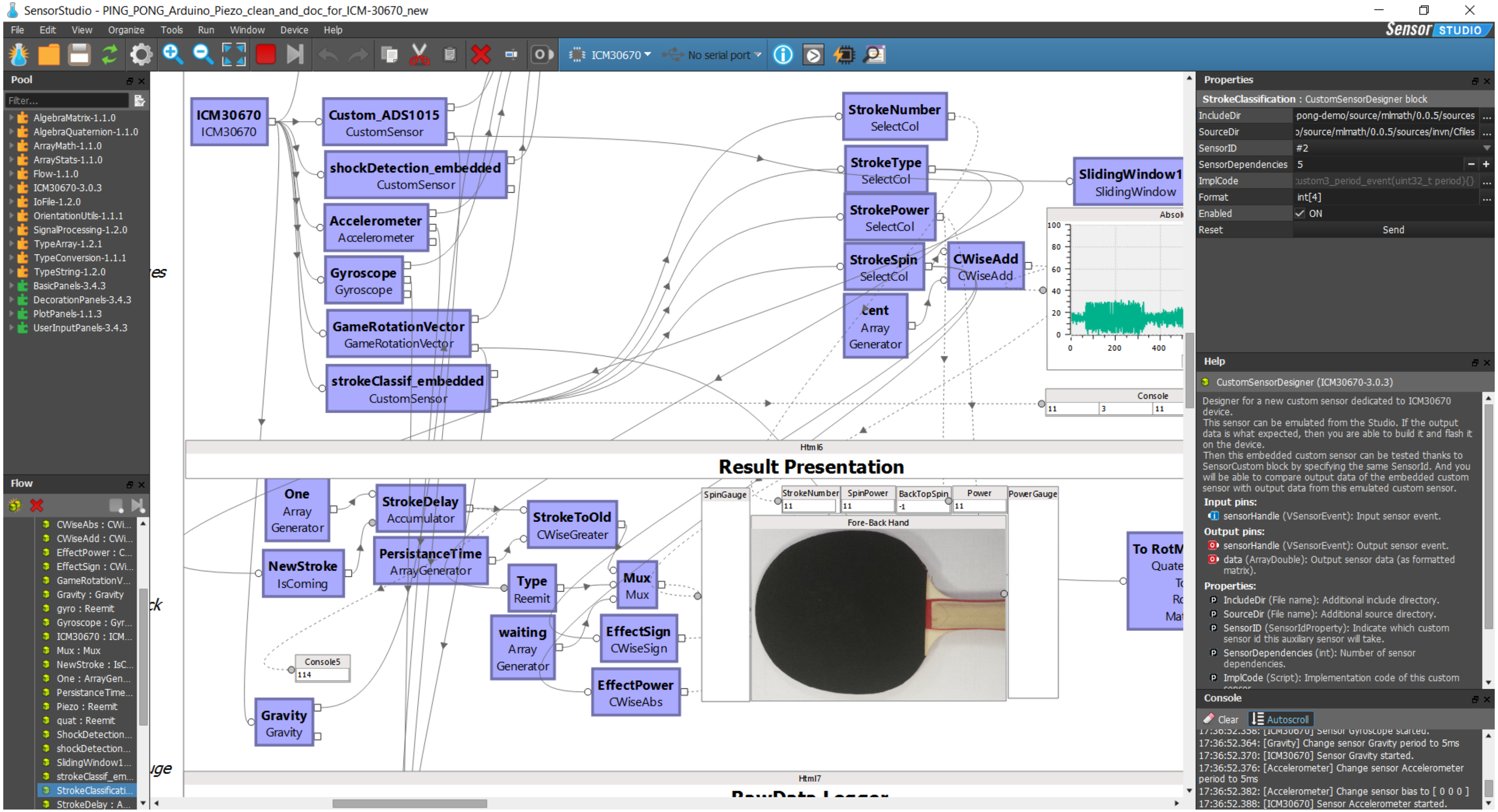
```
Flash started...
Flashing firmware from C:\InvenSense\SensorStudio\2.2.0-SprintD-test3\firefly\devices
\icm30670\3.0.3\build\PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new/bin/
PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670_new.flash.m0.bin...
Flash successful
```

Close

- Outputs: int[3] (Stroke number, Power, Effect)

How: Software test

- You can observe all the algorithm ouputs



Properties

StrokeClassification : CustomSensorDesigner block

IncludeDir	pong-demo/source/mlmath/0.0.5/sources
SourceDir	p/source/mlmath/0.0.5/sources/invn/Cfiles
SensorID	#2
SensorDependencies	5
ImplCode	:ustom3_period_event(uint32_t period){}
Format	int[4]
Enabled	✓ ON
Reset	Send

Help

CustomSensorDesigner (ICM30670-3.0.3)

Designer for a new custom sensor dedicated to ICM30670 device.
This sensor can be emulated from the Studio. If the output data is what expected, then you are able to build it and flash it on the device.
Then this embedded custom sensor can be tested thanks to SensorCustom block by specifying the same SensorId. And you will be able to compare output data of the embedded custom sensor with output data from this emulated custom sensor.

Input pins:

- sensorHandle (VSensorEvent): Input sensor event.

Output pins:

- sensorHandle (VSensorEvent): Output sensor event.
- data (ArrayDouble): Output sensor data (as formatted matrix).

Properties:

- IncludeDir (File name): Additional include directory.
- SourceDir (File name): Additional source directory.
- SensorID (SensorIdProperty): Indicate which custom sensor id this auxiliary sensor will take.
- SensorDependencies (int): Number of sensor dependencies.
- ImplCode (Script): Implementation code of this custom sensor

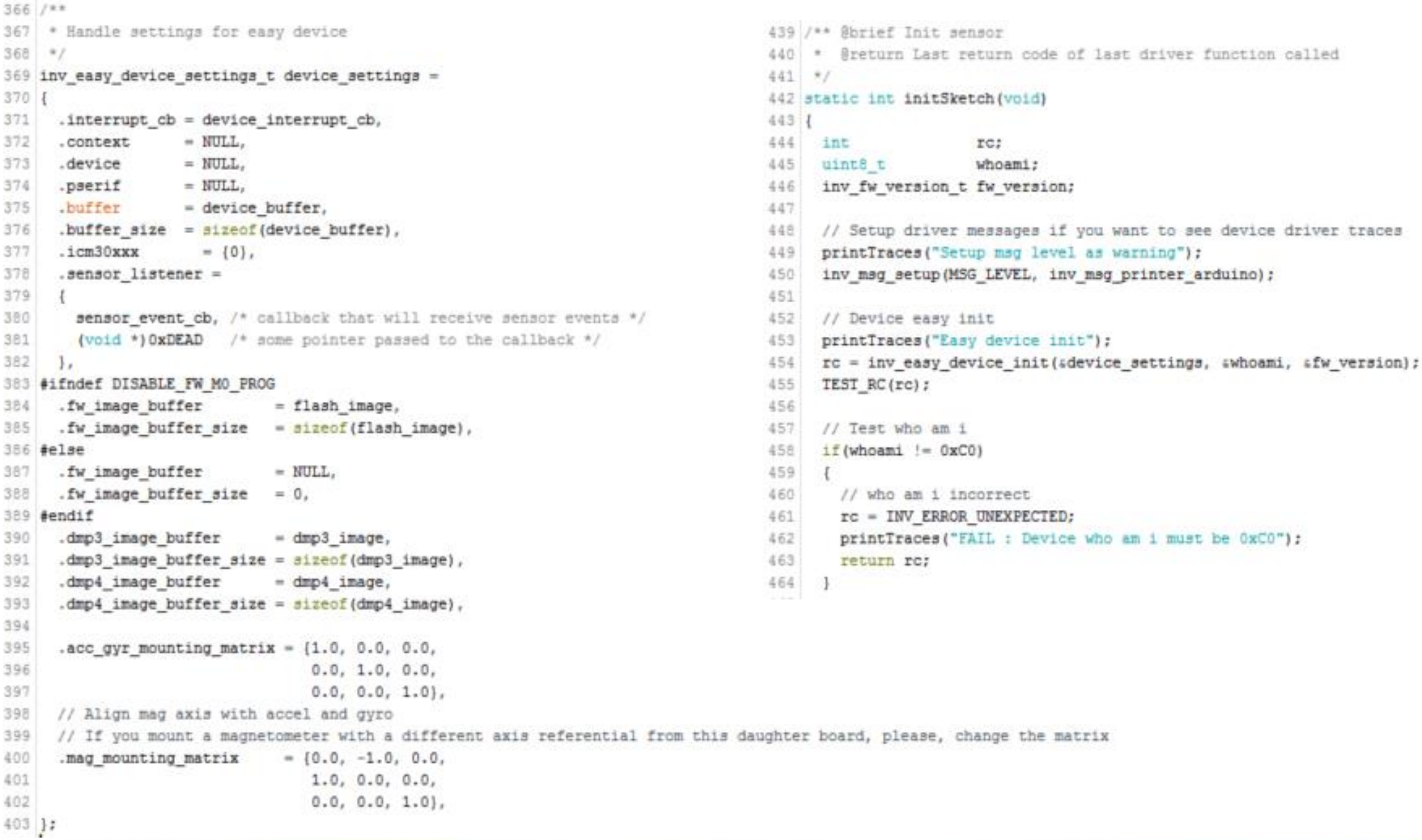
Console

Clear Autoscroll

```
17:36:52.358: [ICM30670] Sensor Gyroscope started.
17:36:52.364: [Gravity] Change sensor Gravity period to 5ms
17:36:52.370: [ICM30670] Sensor Gravity started.
17:36:52.376: [Accelerometer] Change sensor Accelerometer
period to 5ms
17:36:52.382: [Accelerometer] Change sensor bias to [ 0 0 0 ]
17:36:52.388: [ICM30670] Sensor Accelerometer started.
```

How: Software – Arduino init FireFly

- Load&start FireFly

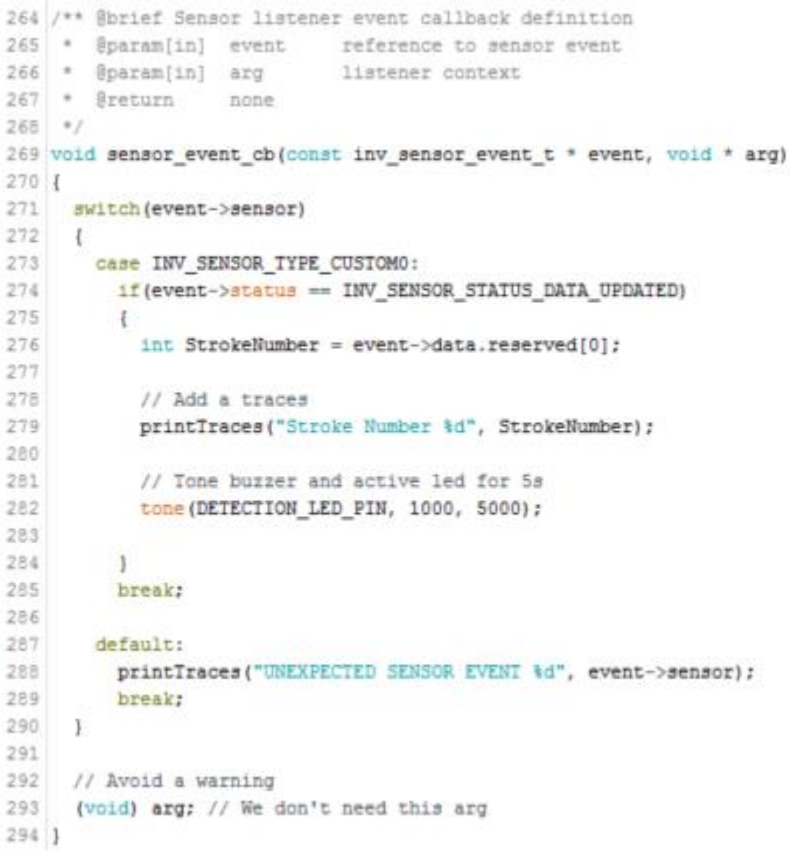


```c
/**
 * Handle settings for easy device
 */
inv_easy_device_settings_t device_settings =
{
  .interrupt_cb = device_interrupt_cb,
  .context       = NULL,
  .device        = NULL,
  .pserif        = NULL,
  .buffer        = device_buffer,
  .buffer_size   = sizeof(device_buffer),
  .icm30xxx        = {0},
  .sensor_listener =
  {
    sensor_event_cb, /* callback that will receive sensor events */
    (void *)0xDEAD   /* some pointer passed to the callback */
  },
#ifndef DISABLE_FW_M0_PROG
  .fw_image_buffer          = flash_image,
  .fw_image_buffer_size     = sizeof(flash_image),
#else
  .fw_image_buffer          = NULL,
  .fw_image_buffer_size     = 0,
#endif
  .dmp3_image_buffer        = dmp3_image,
  .dmp3_image_buffer_size = sizeof(dmp3_image),
  .dmp4_image_buffer        = dmp4_image,
  .dmp4_image_buffer_size = sizeof(dmp4_image),

  .acc_gyr_mounting_matrix = {1.0, 0.0, 0.0,
                              0.0, 1.0, 0.0,
                              0.0, 0.0, 1.0},
  // Align mag axis with accel and gyro
  // If you mount a magnetometer with a different axis referential from this daughter board, please, change the matrix
  .mag_mounting_matrix      = {0.0, -1.0, 0.0,
                              1.0, 0.0, 0.0,
                              0.0, 0.0, 1.0},
};
```

```c
/** @brief Init sensor
 *  @return Last return code of last driver function called
 */
static int initSketch(void)
{
  int               rc;
  uint8_t           whoami;
  inv_fw_version_t fw_version;

  // Setup driver messages if you want to see device driver traces
  printTraces("Setup msg level as warning");
  inv_msg_setup(MSG_LEVEL, inv_msg_printer_arduino);

  // Device easy init
  printTraces("Easy device init");
  rc = inv_easy_device_init(&device_settings, &whoami, &fw_version);
  TEST_RC(rc);

  // Test who am i
  if(whoami != 0xC0)
  {
    // who am i incorrect
    rc = INV_ERROR_UNEXPECTED;
    printTraces("FAIL : Device who am i must be 0xC0");
    return rc;
  }
```

How: Software – Arduino sensor event

- Get Ping Pong data from FireFly ICM-30670 ☺

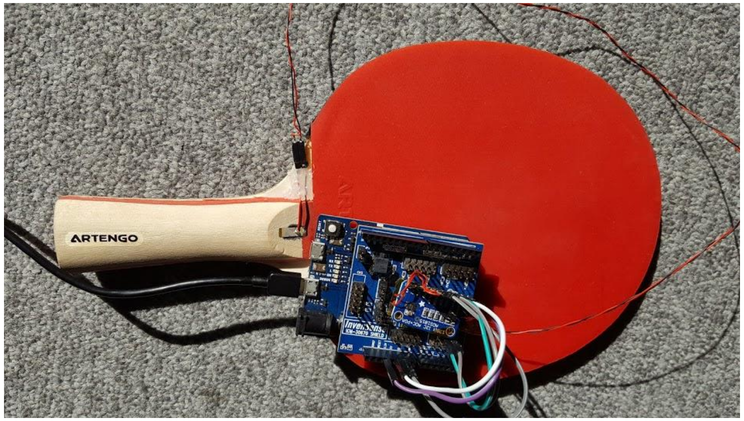


```
/** @brief Sensor listener event callback definition
 *  @param[in]  event     reference to sensor event
 *  @param[in]  arg       listener context
 *  @return     none
 */
void sensor_event_cb(const inv_sensor_event_t * event, void * arg)
{
  switch(event->sensor)
  {
    case INV_SENSOR_TYPE_CUSTOM0:
      if(event->status == INV_SENSOR_STATUS_DATA_UPDATED)
      {
        int StrokeNumber = event->data.reserved[0];

        // Add a traces
        printTraces("Stroke Number %d", StrokeNumber);

        // Tone buzzer and active led for 5s
        tone(DETECTION_LED_PIN, 1000, 5000);

      }
      break;

    default:
      printTraces("UNEXPECTED SENSOR EVENT %d", event->sensor);
      break;
  }

  // Avoid a warning
  (void) arg; // We don't need this arg
}
```

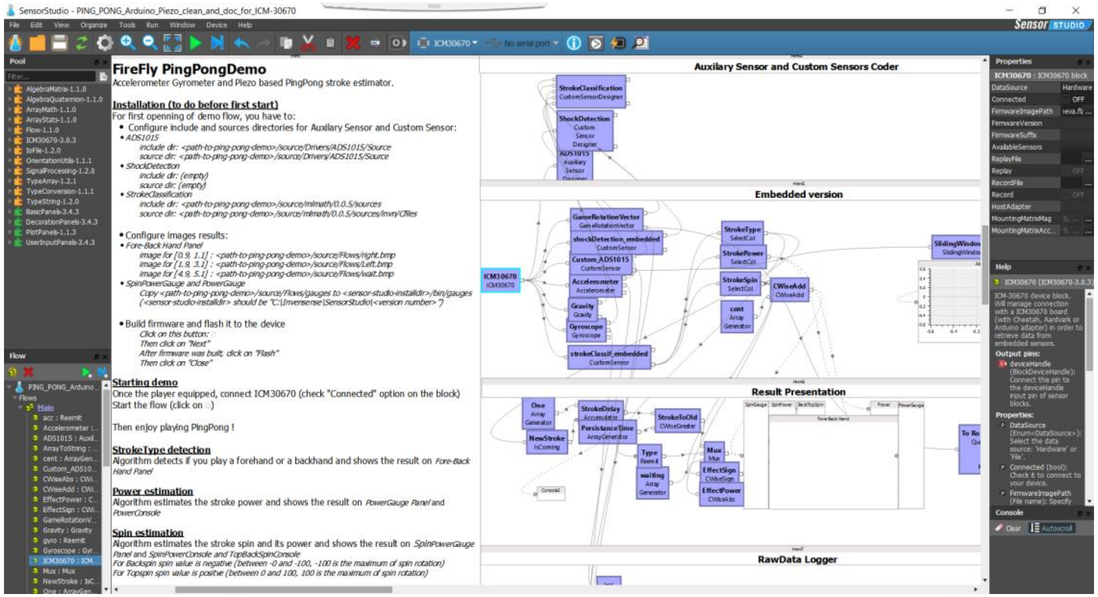
And now?

sensing the FUTURE

- Will try to include Ping Pong in SensorStudio 2.3
- You can build your own
  - Purchase our Development Kits
  - Download SensorStudio



- Use your creativity !


SensorStudio - PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM-30670
File Edit View Organize Tools Run Window Device Help
ICM30670
No serial port
Sensor STUDIO
Pool
Flow
Auxilary Sensor and Custom Sensors Coder
Embedded version
Result Presentation
RawData Logger
Properties
Help
Console
Clear
Autoscroll

FireFly PingPongDemo

Accelerometer Gyrometer and Piezo based PingPong stroke estimator.

Installation (to do before first start)

For first openning of demo flow, you have to:

- Configure include and sources directories for Auxilary Sensor and Custom Sensor:
- *ADS1015*
  include dir: <path-to-ping-pong-demo>/source/Drivers/ADS1015/Source
  source dir: <path-to-ping-pong-demo>/source/Drivers/ADS1015/Source
- *ShockDetection*
  include dir: (empty)
  source dir: (empty)
- *StrokeClassification*
  include dir: <path-to-ping-pong-demo>/source/mlmath/0.0.5/sources
  source dir: <path-to-ping-pong-demo>/source/mlmath/0.0.5/sources/invn/Cfiles

- Configure images results:
- *Fore-Back Hand Panel*
  image for [0.9, 1.1] : <path-to-ping-pong-demo>/source/Flows/right.bmp
  image for [1.9, 3.1] : <path-to-ping-pong-demo>/source/Flows/Left.bmp
  image for [4.9, 5.1] : <path-to-ping-pong-demo>/source/Flows/wait.bmp
- *SpinPowerGauge and PowerGauge*
  Copy <path-to-ping-pong-demo>/source/Flows/gauges to <sensor-studio-installdir>/bin/gauges
  (<sensor-studio-installdir> should be "C:|Invensense|SensorStudio|<version number>")

- Build firmware and flash it to the device
  Click on this button:
  Then click on "Next"
  After firmware was built, click on "Flash"
  Then click on "Close"

Starting demo

Once the player equipped, connect ICM30670 (check "Connected" option on the block)
Start the flow (click on)

Then enjoy playing PingPong !

StrokeType detection

Algorithm detects if you play a forehand or a backhand and shows the result on *Fore-Back Hand Panel*

Power estimation

Algorithm estimates the stroke power and shows the result on *PowerGauge Panel* and *PowerConsole*

Spin estimation

Algorithm estimates the stroke spin and its power and shows the result on *SpinPowerGauge Panel* and *SpinPowerConsole and TopBackSpinConsole*
For Backspin spin value is negative (between -0 and -100, -100 is the maximum of spin rotation)
For Topspin spin value is positive (between 0 and 100, 100 is the maximum of spin rotation)

Thank You