sensing the FUTURE

InvenSense Developers Conference 2016



SensorStudio

Deep Dive





- SensorStudio Introduction
- Sensor Hub Introduction

Agenda

- Understand SensorStudio Samples
- Design your Sensor fusion
- Design your Sensor driver
- Use your CustomSensor
- Make SensorStudio work with your Product

New ICM-20690 Development Kit



SensorStudio ICM-20690 Development Kit

SensorStudio Introduction

- Graphical User Interface
- View into GenericSensorHub
- Extend GSH with your sensors & algorithms!



GenricSensorHub Introduction



- GenricSensorHub and ICM-20690
- Includes InvenSense Motion Algorithms
- Sensor Framework designed for extensibility
 - Add new algorithms
 - Add new sensor drivers



Feature samples

sensing the **FUTURE**

Welcome screen

- guide you to select the appropriate feature samples

👗 SensorStudio			- 0 ×
File Edit. View Organize Tools Run Window Device Help			Sensorstudio
Welcome to SensorStudio			/ Version / Hide 2.2.0-SprintD-test3 / Hide
Create	Open	Learn	
Empty flow	T Browse	i Interactive tutorial	▲ Hardware quick start guide
Project templates	RECENT FILES	? SensorStudio user documentation	
<mark>Х</mark> сян	PING_PONG_Arduino_Piezo_clean_and_doc_for_ICM	Samples & Tutorials	
🔀 ІСМ30670	Piezo_clean_and_doc_for_ICM	GSH basic *	GSH orientation sensors **
	PING_PONG_Arduino_Piezo_clean_and_doc_for_Gene PING_PONG_Arduino_Piezo_clean_and_doc	Discover SensorStudio and your GenericSensorHub evaluation board.	Evaluate GenericSensorHub orientation sensors.
		GSH physical sensors **	GSH auxilliary custom sensor 🛛 🖈 🛧
		Evaluate GenericSensorHub physical sensors.	Create the driver for an auxilliary sensor on the GenericSensorHub board.
		GSH with Sensirion RHT Sensor ★★★	GSH custom sensor ★★★
		Sensirion SHT3x RHT sensor driver on the GenericSensorHub board .	Create your own sensor fusion algorithms and embed them on the GenericSensorHub firmware.
		GSH Record & Replay ★★★	GSH and sensor decimation ★★★★
		Discover how to record data from your device and replay them into your flow	Illustrates how the "decimator" option behaves
		GSH Door opening detection custom sen $\star \star \star \star$	GSH Lightsaber custom sensor ★★★★
		Complex example of a custom sensor implementation detecting opening/closing of a door.	Complex example of a custom sensor implementation simulating lightsaber behavior.
		ICM30670 basic *	ICM30670 orientation sensors **
Follow Us / C Developers forum / M Contact support /			designed by InvenSense

InvenSense Inc. Company Confidential

Orientation sample

- Concrete use of RV, GRV, GeomagRV features
- SensorStudio runs conversion & visualization





InvenSense

Door Opening Detection

- Concrete use of GRV & CustomSensor features
- SensorStudio runs CustomSensor & visualization description





Design your sensor fusion – c

- CustomSensorDesigner skeleton code as a guide
- CustomSensorDesigner code runs on Desktop



sensing the **FUTURE**

Sitor

Sensor Framework integration

notify & subscribe pattern to consume/produce sensor data





InvenSense

sensing the **FUTURE**

InvenSense Inc. Company Confidential

11

Sensor Framework tegration

sensing the **FUTURE**

- TYPE-XSensor_data_event()
 - Called upon receiving a new sensor data of a given type
 - You will have as many TYPE-XSensor_data_event() functions to implement as you have subscribed sensors
 - This is the place to store and process your data
 - Replace TYPE-XSensor_ by other sensor data type
 - When you are done with your processing, use the **notify()** function to pass on your produced data.





InvenSense

DoorOpening sample - data_event()

static void game_rotation_vector_data_event(uint32_t timestamp, void* data, uint16_t len)

```
assert(len == sizeof(VSensorDataQuaternion));
VSensorDataQuaternion* q = (VSensorDataQuaternion*) data;
float angle = 2 *acos(intq30_to_float(q->w)) * 180 / 3.14159;
```

```
// Test angle if angle is > THRESHOLD
if((angle > DOOR_OPENING_DETECTION_THRESHOLD_ANGLE_RISE) &&
  (door_open_detected == false))
{
    // Set flag
    door_open_detected = true;
  }
  else if ((door_open_detected == true) &&
        (angle < DOOR_OPENING_DETECTION_THRESHOLD_ANGLE_DROP))
  {
    door_open_detected = false;
  }
  /* send data event */
  notify(timestamp, &door_open_detected, sizeof(door_open_detectec);
}</pre>
```



InvenSense

Sensor Framework Doc

sensing the **FUTURE**

- C Editor shortcut to Sensor Framework API
- Complete list of
 - Sensor data
 - Sensor functions

available





Design your sensor fusion

- Select your features, assign CustomSensor #IDs
- Cross-build your code for Nucleo
- Flash your Firmware

<i>(</i>)	GSH ▼ • ♀ COM12 ▼ ①	× 🤹 🖈 🔇	Flash	×
Build and flash X	Firmware name GSH-custom-sensor-door-ope Build process	ning	Firmware Select the firmware to flash:	
Firmware name: GSH-custom-sensor-door-opening Version suffix: custom Custom Sensors CustomSensor #0 DoorOpeningDetection Image: CustomSensor #1 CustomSensor #1 Image: CustomSensor #2 CustomSensor #2 Image: CustomSensor #3 Firmware configuration Image: CustomSensor #3 Debuggable firmware Image: Custom Sensor #3	Status from C:\InvenSense\SensorStu \2.2.0\frefly\devices\gsh\1.0.6\build\GSH-c sensor-door-opening\DoorOpeningDetectio from C:\InvenSense\SensorStu \2.2.0\frefly\devices\gsh\1.0.6\build\GSH-c sensor-door-opening\DoorOpeningDetectio C:/InvenSense/SensorStudio/2.2.0\frefly\d 1.0.6\sources/stm32f4x/CMSIS/Core/core_ warning: #warning "Compiler generates FPU for a device without an FPU (checkFPU Wcpp] #warning "Compiler generates FPU ins device without an FPU (checkFPU_PRES 	Success dio ustom- n.inc:1, dio ustom- wrap.c:70: evices/gsh/ crm4.h:131:8: J instructions PRESENT)" [- tructions for a ENT)"	GSH-custom-sensor-door-opening.bin (current build) ▼ Flash Status : Sun Flash started Flashing firmware from C:\InvenSense\SensorStudio \2.2.0\firefl\devices\gsh\1.0.6\build\GSH-custom-sensor- door-opening/bin/GSH-custom-sensor-door-opening.bin Flash successful	ccess
Next	Config	Flash		lose

sensing the

FUTURE

Running your sensor fusion

- CustomSensor runs embedded
 - Firmware loaded, GenericSensorHub Connected
 - GRV & CustomSensor Enabled



Debug your code in Eclipse

- Launch Eclipse project
- Set your breakpoint & Step in your code



Migrating to a Full Prototype

sensing the

Simple and fast migration from Algo/SW development --> proto/product development





A - Tested and validated project is integrated onto Nucleo via SensorStudio

B - Create new Application code to include rest of the platform needs i.e . connectivity shield



SW Development Environment for GenericSensorHub APP/HW product development Environment

Build your app in your IDE

- sensing the **FUTURE**
- Curstomer Application Task to build your app
- Extend the sample code provided
 get all outputs from GenericSensorHub
- Add your display, wireless, peripherals, ...



Design your sensor fusion - Summary

- Add a CustomSensorDesigner block
- Open CustomSensorDesigner C Editor
- Implement the Sensor Framework interface
 - Skeleton code provided to simplify programming tasks
 - Sample codes are included too
- Select the features you want to use
- Build your code for GSH MCU
- Flash the Nucleo
- Add a CustomSensor block `
- Select the CustomSensor ID
- Observe GSH outputs
- Debug with Eclipse
- Build your app





InvenSense

sensing the

FUTURE

Design your sensor driver – C Editor

- AuxiliarySensorDesigner skeleton code as a guide
- AuxiliarySensorDesigner runs on HW (only)



sensing the

Sensor Framework Doc

sensing the FUTŬRE

- C Editor shortcut to Sensor Framework API
- Complete list of functions **InvenSense**
 - auxiliary I2C bus
 - system timer
 - tasks

AuxiliarySensorDesigner.ImplCode - C Editor



Showing how to implement an auxiliary sensor by using I2C but

Generic S	ensor	Hub
firmware	1.1.0	

Documentation of the Generic Sensor Hub firmware running on Nucleo Board

Main Page	Related Pages	Modules	Classes	Files	Q* Search
Extension	framework		, ,		
Auxilia	ry bus APIs	int installant		al man / inc. also to acc	· 12a muna A aunu 12a
INV	_AUX_I2C_NUM_0	int inv_snex	(Laux_I2C_re	ad_reg (inv_snext_au)	C_I2C_num_t aux_I2C,
inv	_shext_aux_i2c_nun			uint8_t	addr,
inv	_shext_aux_i2c_cloc			uint8_t	reg,
inv	_shext_aux_i2c_init			uint8_t *	data,
inv_	_shext_aux_i2c_rea			uint16_t	len
inv	_shext_aux_i2c_writ)	
Auxial	ary GPIO APIs			,	
► inv_	_shext_gpio_struct_1	Dead registe	r over Auvilian	/ I2C bue	
INV	GPIO_PIN5	Read registe		/ 120 bus.	
inv_	_shext_gpio_port_t	Reading a re	egister consists	in an I2C Write of 1 byt	es (register address) followed by an
	_snext_gpio_mode_	I2C read of N	V bytes.		
inv_	shext_gpio_clear	Morning			
inv_	shext_gpio_get_sta	vvarning It is po	noccible to re	ad more than 16 bytes r	per request
inv	shext_gpio_init		possible to re	ad more than to bytes p	ber request
inv	shext apio_struct i	Parameters			
inv	shext gpio toggle	[in]	aux_i2c I2C	auxiliary bus base adres	ss register to use
Sema	ohore APIs	[in]	addr 12C	peripheral 7bits address	8
🔻 Task A	Pls	[in]	rea rea	ster to read	
▶ inv	_shext_task	[out	ldata noir	ater to data buffer	
inv	shext_task_t	lout	Juata poli		
		lini	ien nun	iber of data to read (mus	SI DE <= 16)

Sensor Framework integration



• Task & notify pattern to produce sensor data



Sensor Framework integration

• init()

 Called once, Initialize the CustomSensor state (after the GSH device is powered up, when Sensor Framework is started: GSH "Connected" property)



InvenSense Inc. Company Confidential

sensing the

FUTŬRE

Sensor Driver - Define a task

• Responsible for polling data from your sensor at the specified Period

```
/**
 * Code for the custom task that will retrieve data from the AK09911
 */
static void MyCustomTaskCode(void * arg)
{
    /* get current system time in us */
    uint32_t t = inv_shext_get_systick();
    int16_t sample[3];
    ak09911_get_sample(sample); /* retrieve sensor data */
    notify(t, sample, sizeof(sample)); /* notify data to the outside world */
```

```
/* The HW sensor performs a single acquisition, so ask for another one */
ak09911_start_single();
```

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

Note : the notify is here, you don't have to implement a data_event(), this is the difference with an algorithm. Sensor Driver is only producing data while an algorithm consumes data from depending sensors & produces data.

sensing the

FUTŬRE

AuxiliarySensorDesigner - init()

Create your task & initialize l2C at init()
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 AK09911 sensor (should fail if sensor is not connected) */
if(ak09911_init() != 0) return -1;

/* initialize task object that is used in this sample */
inv_shext_task_create(&MyCustomTask, MyCustomTaskCode,

0 /* optional pointer passed to MyCustomTaskCode() */);





InvenSense

Sensor Framework integration

• period_event()

- Called just before your custom sensor is started.
- Allows to configure the frequency of your CustomTask

subscribe_event()

Allows to start your CustomTask



AuxiliarySensorDesigner - period_event()

sensing the **FUTURE**

• Set your task period at period_event()

```
static void period_event(uint32_t period) {
   /* In this example we update the period of our task */
   inv_shext_task_set_period(&MyCustomTask, period);
```

Start your task at subscribe_event()

```
static void subscribe_event() {
    /* Start HW sensor and ask for a data acquisition */
    ak09911_start_single();
```

/* In this example we simply start the task that was created in init() function. $^{\prime /}$



Design your Sensor Driver

- Select your features, assign CustomSensor #IDs
- Cross-build your code for Nucleo
- Flash your Firmware

^	ssh ▼ • Com12 ▼	1 🔊 🕭		
K S			Ŋ	
Build and flash ? X	Build and flash	? ×	E Flash	? ×
Firmware name yag Version suffix (required, up to 7 alphanumeric characters) test8 Custom Sensors New Custom Sensors have been detected. They will be added to the new firmware. ✓ CustomSensor #0 YAS533 ✓ CustomSensor #1 ~ ✓ CustomSensor #2 ~ ✓ CustomSensor #3 ~	Firmware name yas Build process Status Build started Configuring files Building firmware Build successful Firmware size: 50516 bytes	Success	Firefly Firmware Select the Firmware to flash C:/InvenSense/InvenSenseS Flash Status	itudio/2.0.0-test1/firef 🔻 🛄
Optional Sensors The following sensors will be added to the new firmware. ✓ Compass Gravity Gravity Linear acceleration Orientation	Config	Flash		Flash

sensing the

FUTURE

Design your Sensor Driver - Summary

- Add an AuxiliarySensorDesigner block
- Open C Editor
- Implement the Sensor Framework interface

 Sample code is provided to simplify programming tasks
- Connect it to GSH block
- Build your code for GSH
- Flash the Nucleo
- Add a CustomSensor block
- Select the CustomSensor ID
- Observer GSH outputs

GSH

GSH



Close

X

Errors (0)

Apply

sensing the

AuxiliarvSensorDesigner.ImplCode - C Editor

Auxiliary sensor sample.

* Showing how to implement an auxiliary sensor by using

lict accessing the senso

to communicate with the 3rd

is sample, you must not start the Magnetome

11 sensor is already supported in the built-in FW

API Documentation

AuxiliarySensorDesigner

AuxiliarySensorDesigner

CustomSenso

CustomSensor

* This example

Working with your final Product sensing the **FUTURE**

- SensorStudio can work with your final product too
- "DynamicProtocol" code connect SensorStudio & GenericSensorHub
- "DynamicProtocol" source code is available for you to modify if needed



Presentation Summary



- Learned about SensorStudio
- Learned about GenerciSensorHub
- Evaluated INVN sensor, fusion & hub
- Understood SensorStudio Samples
- Designed your Sensor fusion
- Designed your Sensor driver
- Used your CustomSensor
- Used SensorStudio with your product



Thank You

