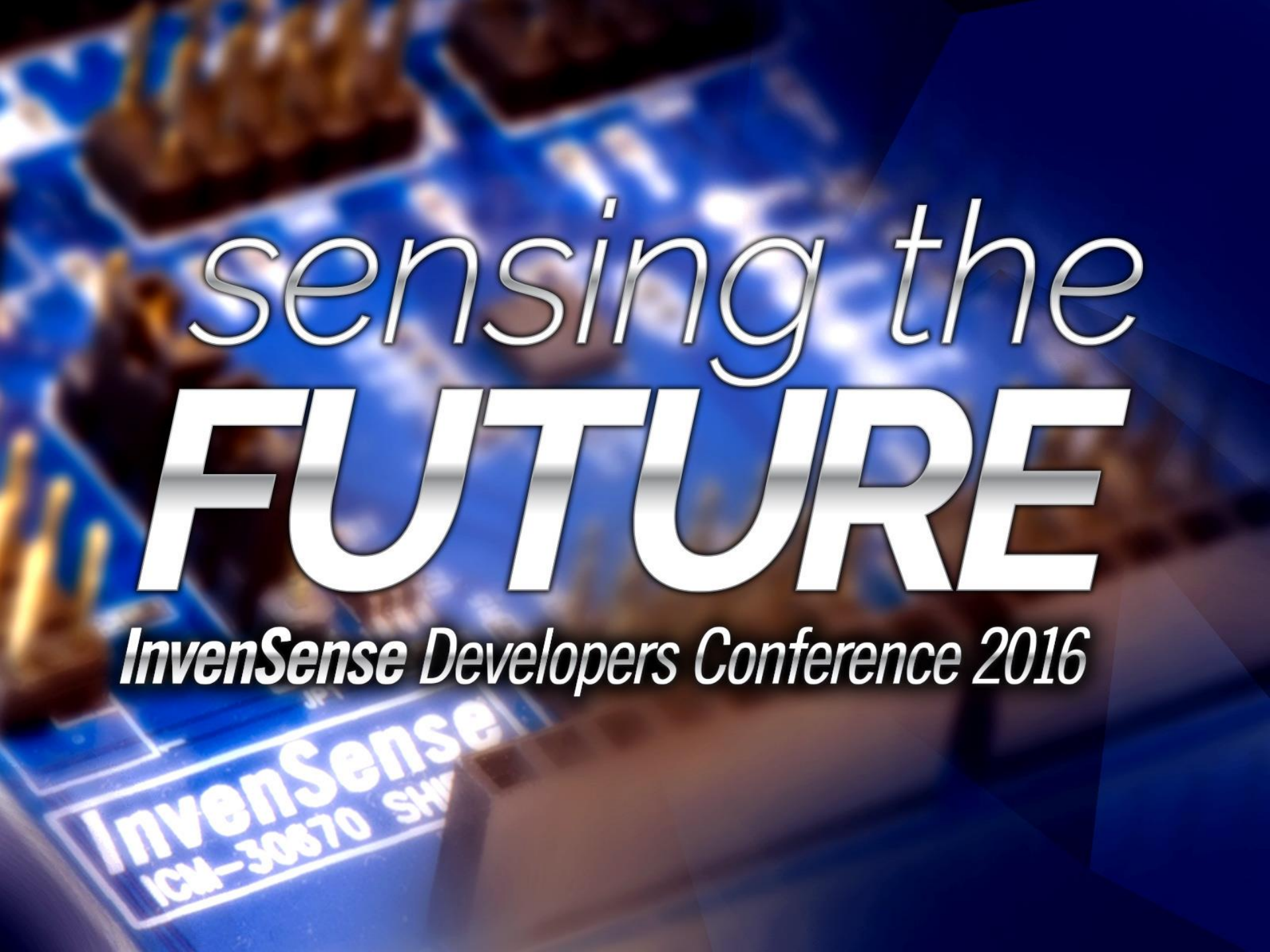

sensing the
FUTURE
InvenSense Developers Conference 2016
InvenSense
ICM-30670

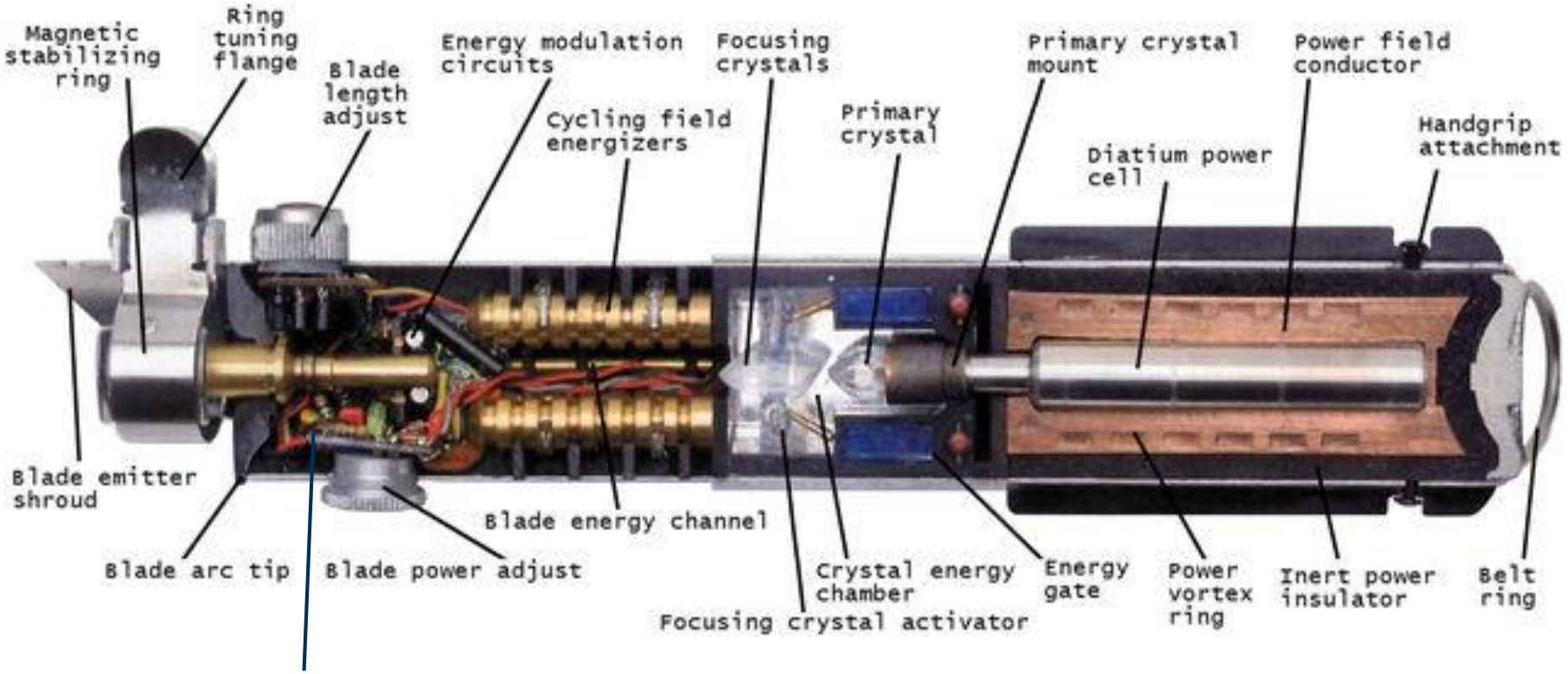
SensorStudio Real Use Cases

Lightsaber “how to“

Joke of the day ☺

sensing the FUTURE

- We have built this for fun !

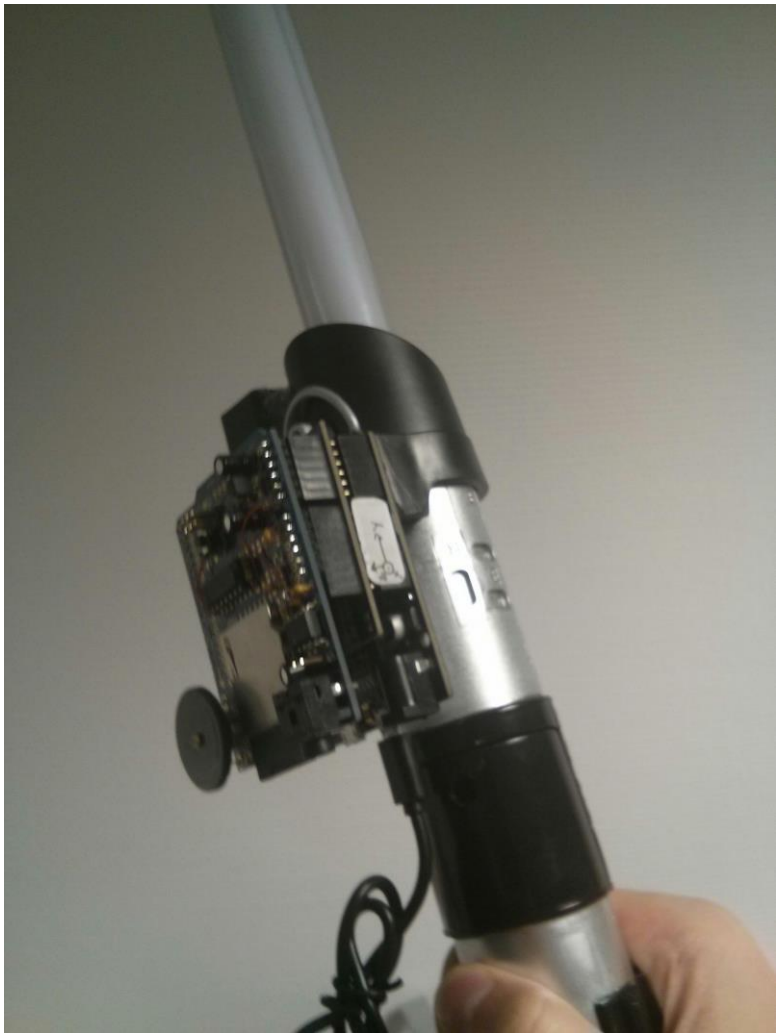


InvenSense **FireFly ICM-30670**

(Source http://starwars.wikia.com/wiki/File:Lightsaber-cutaway.jpg)

Seriously ☺

- We have built this for fun !



Agenda

sensing the FUTURE

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

Why

- We had to test SensorStudio&FireFly programming
- We were looking for a “WOW factor” @ CES’2016
  (extending feature set from the existing toys)

What

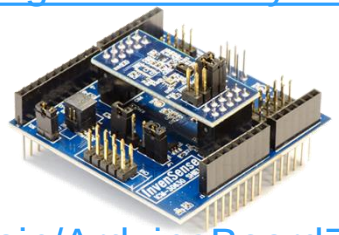
- FireFly ICM-30670
  - Detects Shocks
  - Detects Up/Down/Right/Left/Diagonals
  - Computes Gesture's power
- Arduino
  - Drives the Audio

  (based on FireFly outputs)
- SensorStudio
  - Design/Debug/Demo

How: Hardware BOM

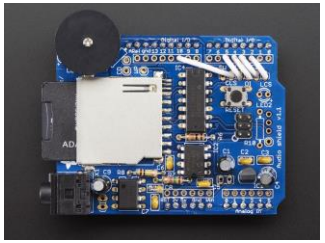
sensing the FUTURE

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



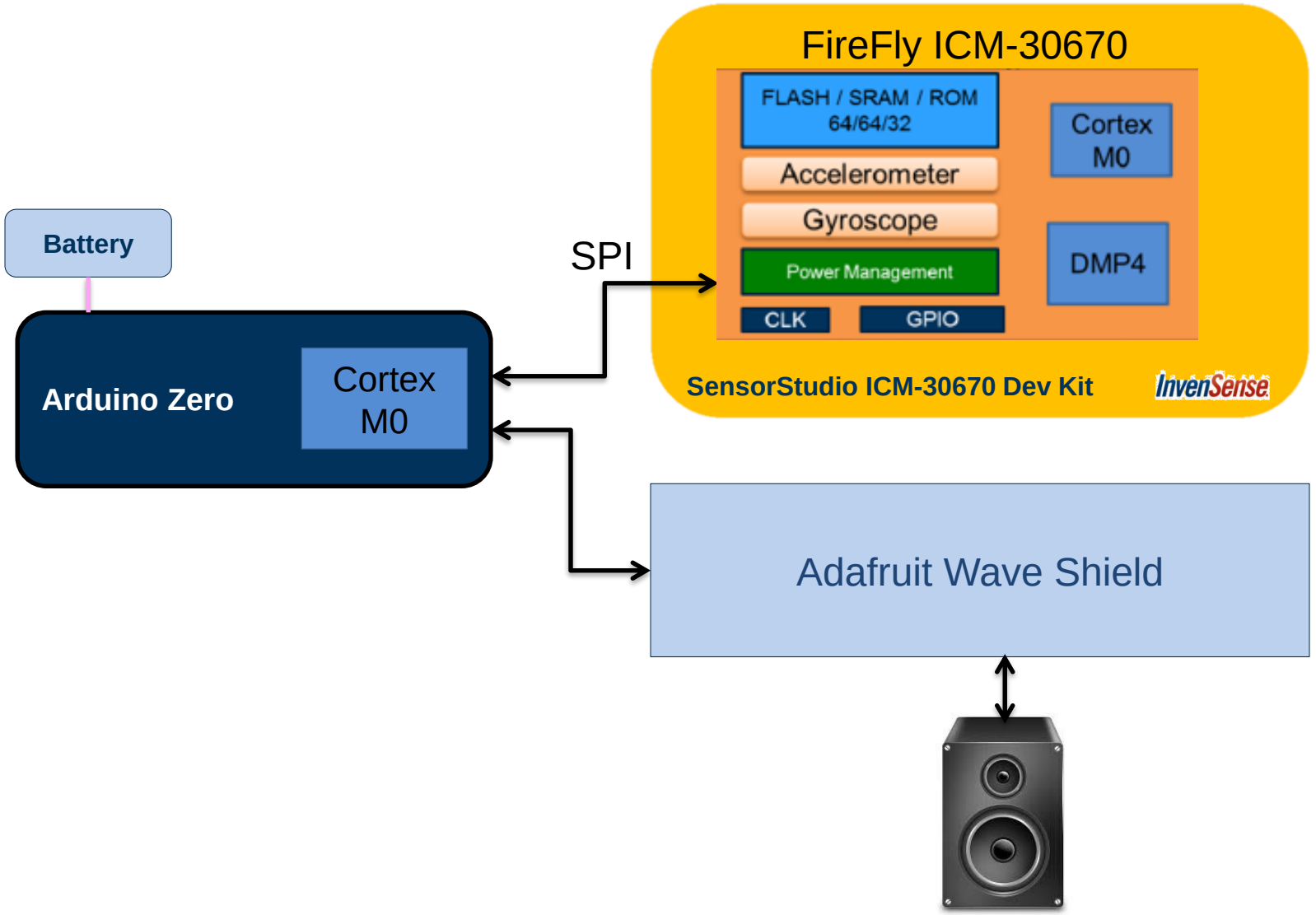
- Arduino Zero - https://www.arduino.cc/en/Main/ArduinoBoardZero

- Adafruit Wave Shield - https://www.adafruit.com/products/94



- Lightsaber - https://www.amazon.com/Anakin-Change-Lightsaber-Discontinued-manufacturer/dp/B00CFWWD7Y

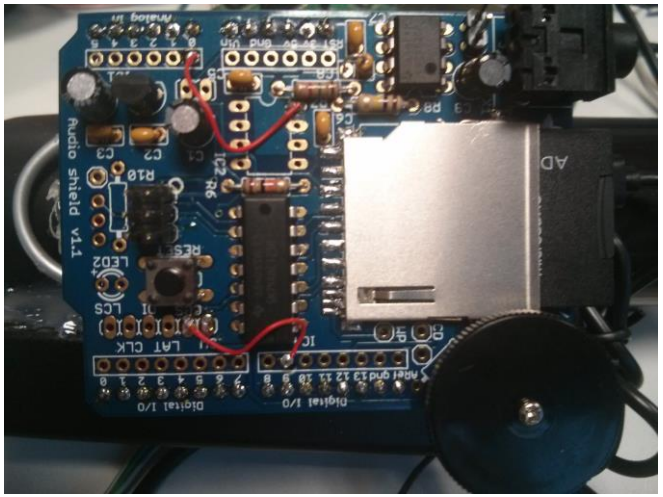
How: Hardware Schematic



How: Hardware Wave adaptations

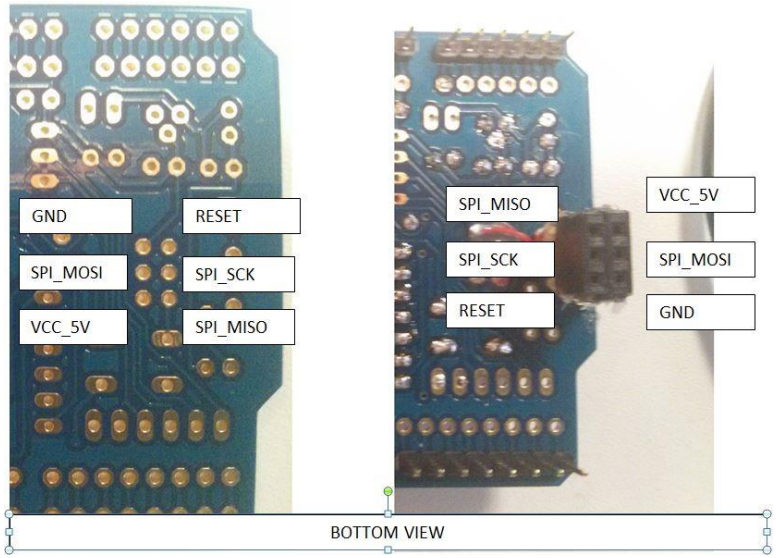
sensing the FUTURE

- Adafruit Wave shield needed modifications



JP9.1 (A0) is wired to IC2.8 (DACA)
JP13.5 (MMC_CS) is wired to JP2.7 (D9)

Also, IC2 & JP13 are not mounted

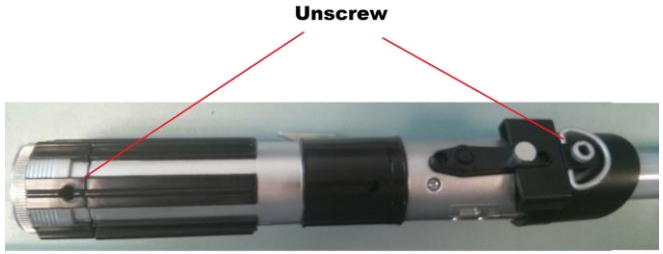


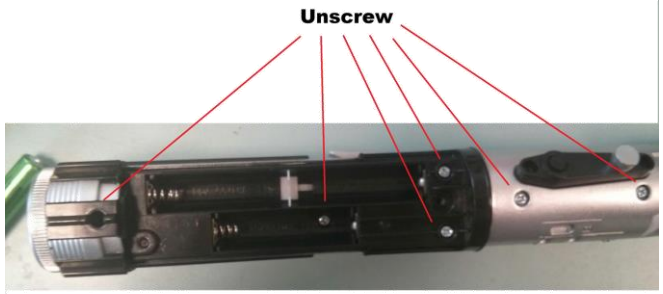
ICSP connector is misplaced and pinout does not correspond to Arduino Zero
You have to glue a female connector at the right place & wire it to the ISCP connector.

- We followed guide @ https://learn.adafruit.com/adafruit-wave-shield-audio-shield-for-arduino/solder

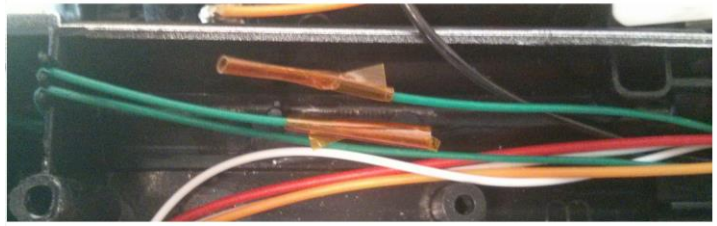
How: Hardware Hack

- Lightsaber toy came with basic motion detection
- We took this out





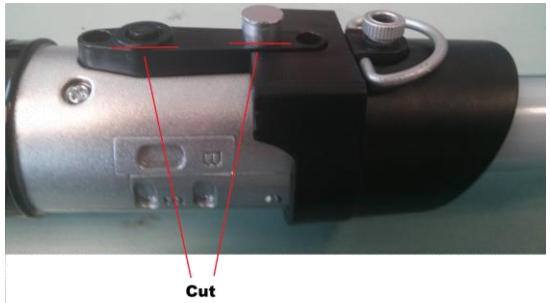
Cut a speaker wire (green)



How: Mechanical hacks

sensing the FUTURE

- **Create a flat surface**

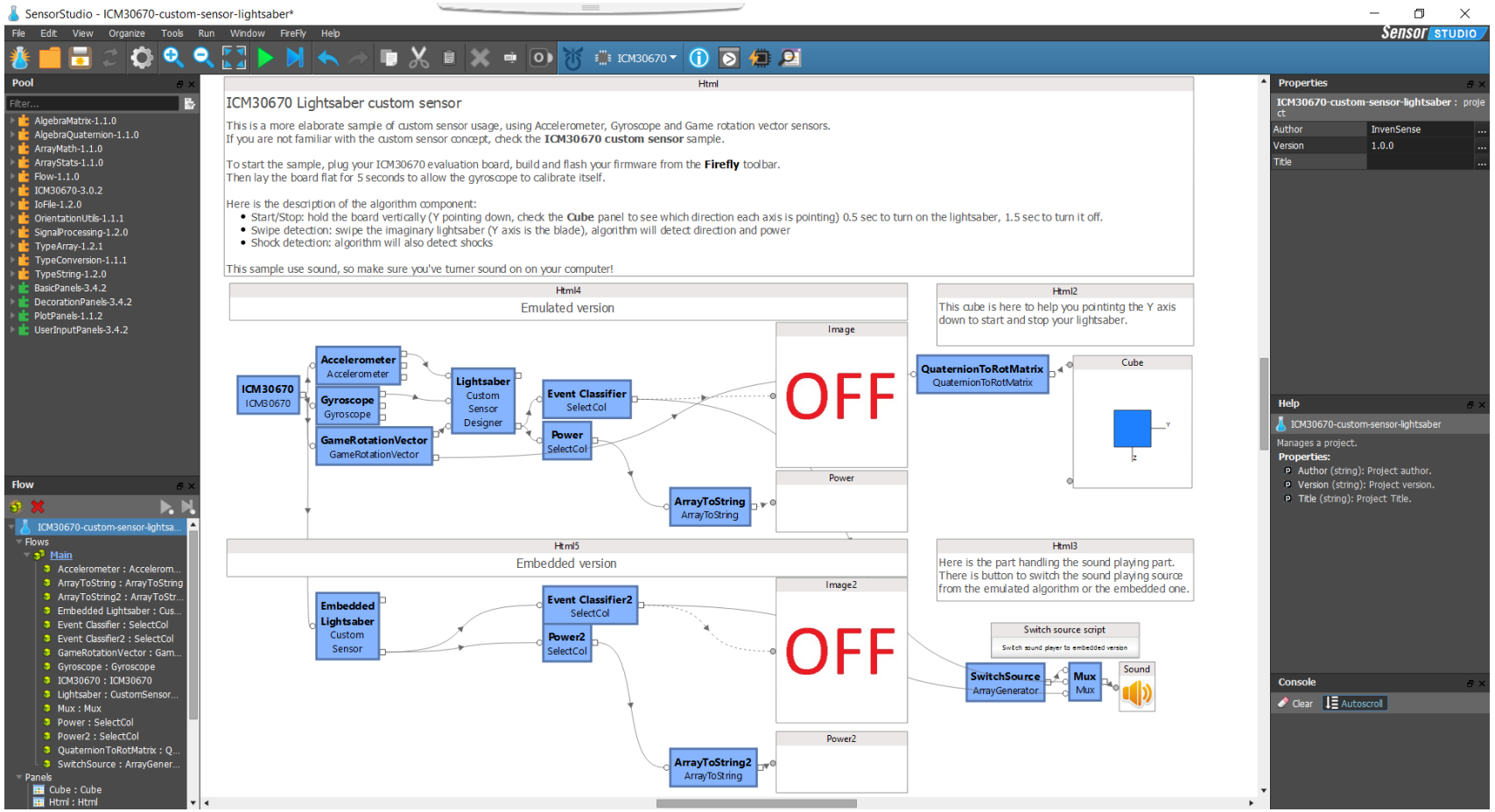


- **Glue Arduino & Battery (need a lot of glue!)**
  USB connectors need to be at the bottom

How: Software SensorStudio

sensing the FUTURE

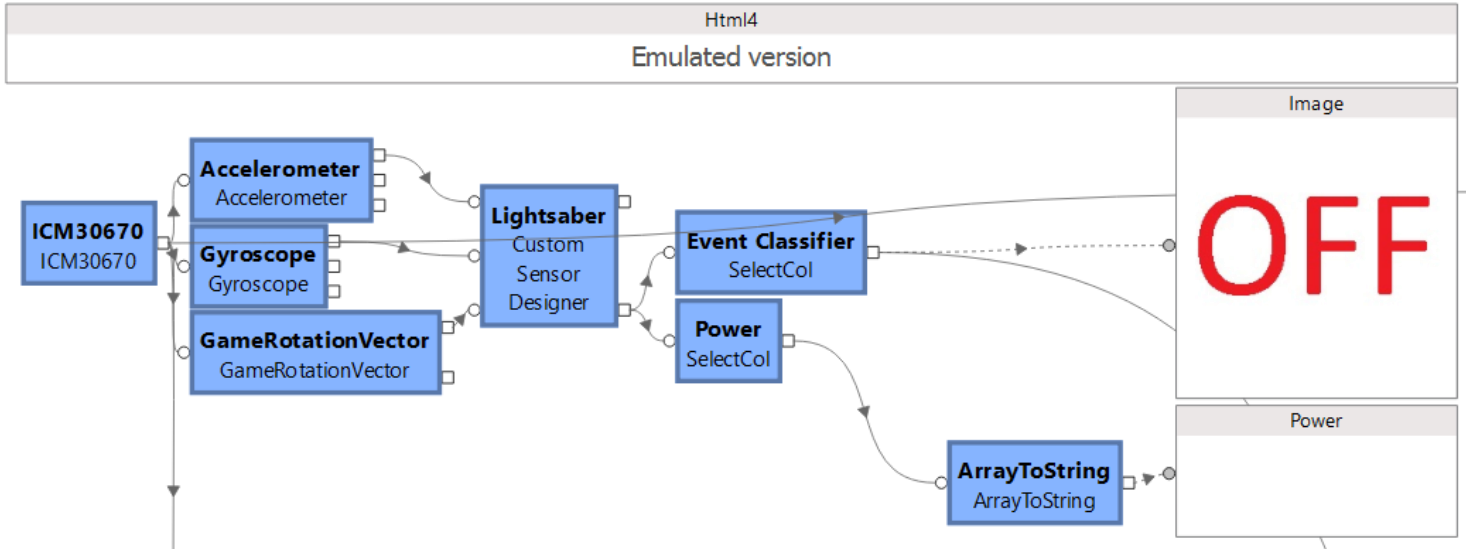
- SensorStudio used to create algorithm (CustomSensor)
- Visualization of algorithm outputs

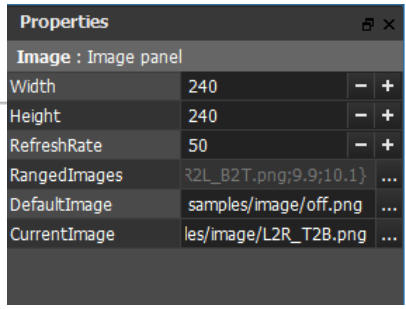


How: Software desktop run

sensing the FUTURE

- Lightsaber algorithm fusing Acc/Gyro/GRV





- Outputs: u8 u8 (Event Classifier, Power)
- 1 image per classification
- Power

How: Software - Shock Detection

sensing the FUTURE

- Principle : Accelerometer 2nd Derivative

```
493     //Check Shock
494     if(AlgoState == Started)
495     {
496         int iAcc1 = (iAcc+1) % 2;
497         AccMoment[0] = buffAcc[iAcc*4+1] - buffAcc[iAcc1*4+1];
498         AccMoment[1] = buffAcc[iAcc*4+3] - buffAcc[iAcc1*4+3];
499         ddAccXZNormSqr = (AccMoment[2] - AccMoment[0])*(AccMoment[2] - AccMoment[0]) + (AccMoment[3] - AccMoment[1])*(AccMoment[3] - AccMoment[1]);
500         if(ShockState == ShockWait)
501         {
502             if(ddAccXZNormSqr > SHOCK_THR)
503             {
504                 ShockState = Shock;
505
506                 if((buffGyroNormSqr[9] - SHOCK_POWER_OFF)>0)
507                     PowerSqr = (buffGyroNormSqr[9] - SHOCK_POWER_OFF);
508                 else PowerSqr = 1;
509                 if (arm_sqrt_f32(PowerSqr, &sqrtRes) == ARM_MATH_SUCCESS)
510                 {
511                     Power = (.5+POWER_SHOCK_GAIN * sqrtRes/(ANALYS_SAMPLES));
512                     if (Power > 100.0)
513                     {
514                         Power = 100.0;
515                     }
516                     else if (Power < 1.0)
517                     {
518                         Power = 1.0;
519                     }
520                 }
521                 else
522                 {
523                     Power = 125.0;
524                 }
525                 notify_result(SABER_SHOCK, (uint8_t)Power);
```

How: Software – Swipes detection

sensing the FUTURE

- Principle : Gyro Norm on yaw/pitch axis > threshold

```
257         //Check Swipe Gyro
258         GyroXZNormSqr = buffGyro[iGyro*4+1]*buffGyro[iGyro*4+1] + buffGyro[iGyro*4+3]*buffGyro[iGyro*4+3];
259         for(i=0;i<9;i++)
260                                 buffGyroNormSqr[i] = buffGyroNormSqr[i+1];
261                         buffGyroNormSqr[9] = (GyroXZNormSqr + FILTER_GYRO_NORM*buffGyroNormSqr[9])/(1.+FILTER_GYRO_NORM);
262
263                         if(GyroXZNormSqr > SWIPE_THR)
264         {
265             iAnalys = 0;
266             GyroAccu[0] = 0;
267             GyroAccu[1] = 0;
268             SwipeState = SwipeAnalys;
269
270             //ReInit swipe Result
271             SwipeRes = No;
272             SwipeHSign = 0;
273             SwipeVSign = 0;
274         }
```

How: Software – Swipes direction

sensing the FUTURE

- Principle : 25 samples where Gyro Norm > threshold

```
277             if(SwipeState == SwipeAnalys)
278             {
279                 iAnalys++;
280 
281 
282                 GyroAccu[0] += GyroEarth[0];
283                 GyroAccu[1] += GyroEarth[1];
284                 if(iAnalys >= ANALYS_SAMPLES)
285                 {
286                     if(fabs(GyroAccu[1]) > SWIPE_HV_THR * fabs(GyroAccu[0]))
287                         SwipeRes = SwipeV;
288                     else if(fabs(GyroAccu[1]) * SWIPE_HV_THR < fabs(GyroAccu[0]))
289                         SwipeRes = SwipeH;
290                     else SwipeRes = SwipeD;
291 
292                     SwipeVSign = 0;
293                     SwipeHSign = 0;
294                     if(SwipeRes == SwipeH || SwipeRes == SwipeD) // = not vertical
295                     {
296                         if(GyroAccu[0] >= SWIPE_SIGN_THR)
297                             SwipeHSign = 1;
298                         else if(GyroAccu[0] < -SWIPE_SIGN_THR)
299                             SwipeHSign = -1;
300 
301                     }
302 
303                     if(SwipeRes == SwipeV || SwipeRes == SwipeD)
304                     {
305                         if(GyroAccu[1] >= SWIPE_SIGN_THR)
306                             SwipeVSign = 1;
307                         else if(GyroAccu[1] < -SWIPE_SIGN_THR)
308                             SwipeVSign = -1;
309 
310                     }
311
```

How: Software – Gestures' power

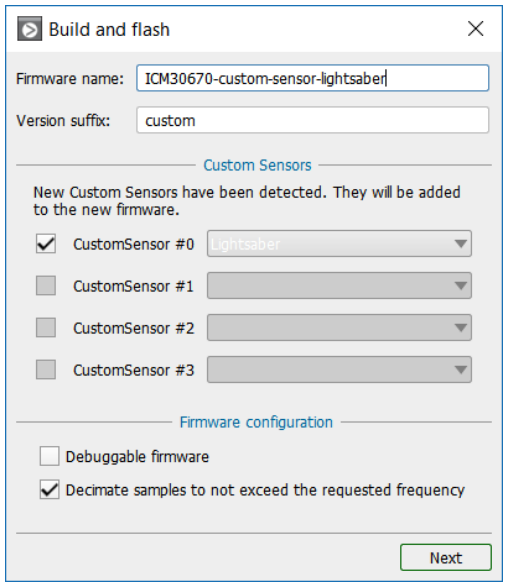
sensing the FUTURE

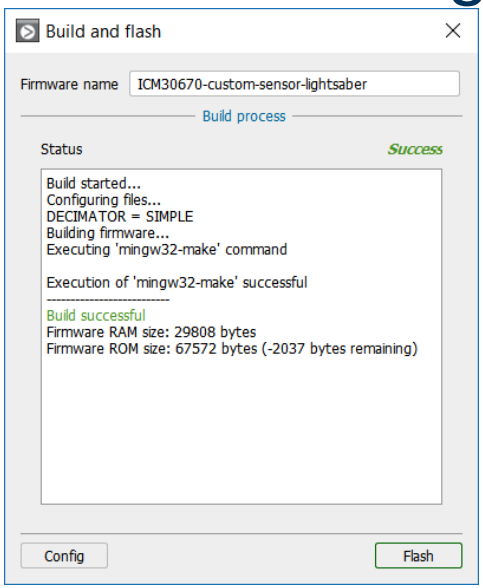
- Principle :
  - Gyro Norm on 25 samples during high speed motion

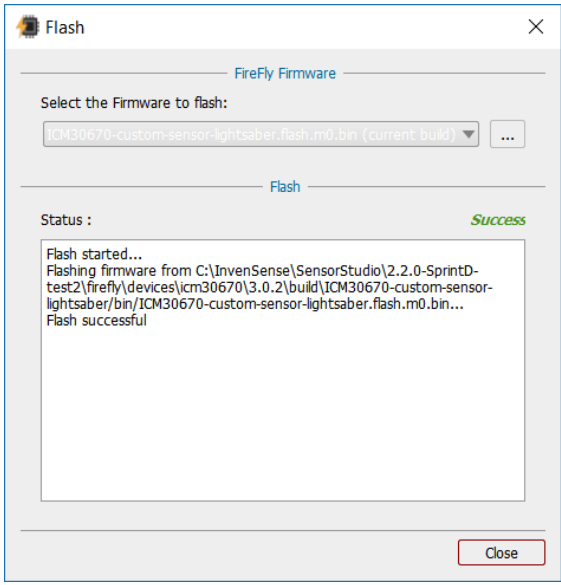
```
312                 if((GyroAccu[0] * GyroAccu[0] + GyroAccu[1] * GyroAccu[1] - SWIPE_THR*ANALYS_SAMPLES)>0)
313                     PowerSqr = (GyroAccu[0] * GyroAccu[0] + GyroAccu[1] * GyroAccu[1] - SWIPE_THR*ANALYS_SAMPLES);
314                 else PowerSqr = 1;
315                 if (arm_sqrt_f32(PowerSqr, &sqrtRes) == ARM_MATH_SUCCESS)
316                 {
317                     Power = (.5+POWER_GAIN * sqrtRes/(ANALYS_SAMPLES));
318                     if (Power > 100.0)
319                     {
320                         Power = 100.0;
321                     }
322                     else if (Power < 1.0)
323                     {
324                         Power = 1.0;
325                     }
326                 }
```

How: Software – Build/Flash FireFly

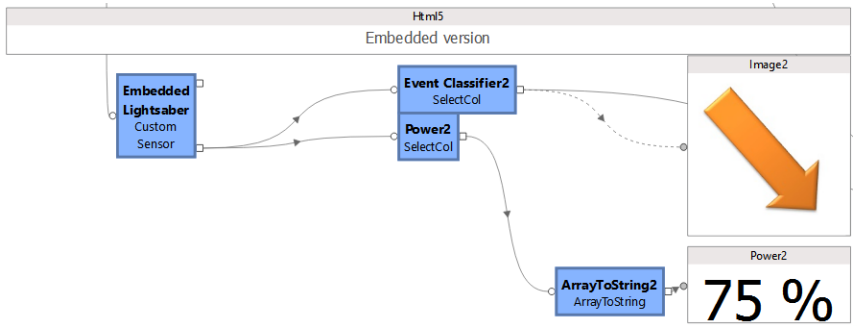
- Add the CustomSensor to FW Configuration
- Build
- Flash







- Embedded version runs



How: Software – Arduino init FireFly

sensing the FUTURE

- Load&start FireFly, get CustomSensor events

```
/**
 * 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_MO_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 b
  .mag_mounting_matrix     = {0.0, -1.0, 0.0,
                              1.0, 0.0, 0.0,
                              0.0, 0.0, 1.0},
};
```

```
/** @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 ir
    rc = INV_ERROR
    printTraces("I
    return rc;
  }
```

```
/** @brief Sensor listener event callback definition
 *  @param[in]  event     reference to sensor event
 *  @param[in]  arg       listener context
 *  @return     none
 */
static 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)
      {
        memcpy(&algoOutput, event->data.reserved, sizeof(algoOutput));

        newAlgoEvent = true;
```

How: Software – Arduino ctrl Wave

sensing the FUTURE

- Make some noise ☺

```
572 /** @brief Arduino sketch loop function
573  *  called in loop by Arduino sketch
574  *  @return     never
575  */
576 void loop()
577 {
578   // Test if there is any char in buffer
579   if(SERIAL_TRACES.available() == 0)
580   {
581     // No, test if interrupt occured
582     if(icm_interrupt_occured != false)
583     {
584       printTraces("interrupt occured");
585
586       // Reset interrupt flag
587       icm_interrupt_occured = false;
588
589       InvAudioZero.stop();
590       waveFinished = true;
591
592       // Interrupt occur poll device
593       inv_device_poll(device_settings.device);
594
595       if (newAlgoEvent == true)
596       {
597         switch (algoOutput.classifier_evt)
598         {
599           case SABER_ON:
600             SaberOn = true;
601             inv_play_wave(FILE_WAVE_SABER_ON);
602             break;
603           case SABER_OFF:
604             SaberOn = false;
605             inv_play_wave(FILE_WAVE_SABER_OFF);
606             break;
```

```
607       // Saber Hit
608       case SABER_SHOCK:
609         inv_play_wave(fileWaveShock[FileShockIdx]);
610         FileShockIdx++;
611         if ( FileShockIdx == NB_SHOCK)
612         {
613           FileShockIdx = 0;
614         }
615         // wait end of play
616         while (waveFinished == false);
617         break;
618       // Swing
619       case SABER_SWING_TOP2BOT:
620       case SABER_SWING_BOT2TOP:
621       case SABER_SWING_L2R_TOP2BOT:
622       case SABER_SWING_L2R:
623       case SABER_SWING_L2R_BOT2TOP:
624       case SABER_SWING_R2L_TOP2BOT:
625       case SABER_SWING_R2L:
626       case SABER_SWING_R2L_BOT2TOP:
627         if (algoOutput.power > 50)
628         {
629           inv_play_wave(fileWaveSwingF[FileSwingFIdx]);
630           FileSwingFIdx++;
631           if ( FileSwingFIdx == NB_SWING_F)
632           {
633             FileSwingFIdx = 0;
634           }
635         }
636         else
637         {
638           inv_play_wave(fileWaveSwingN[FileSwingNIdx]);
639           FileSwingNIdx++;
640           if ( FileSwingNIdx == NB_SWING_N)
641           {
642             FileSwingNIdx = 0;
643           }
644         }
645         break;
```

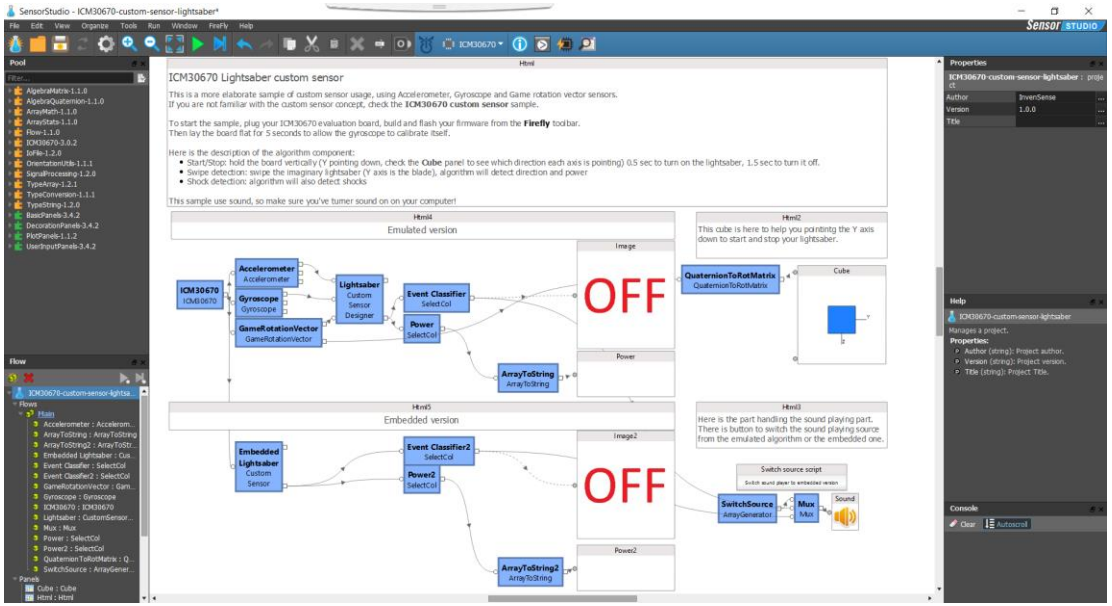
```
533 // Play a wave (blocking)
534 void inv_play_wave(const char *fileName)
535 {
536   File fileHandle;
537
538   waveFinished = false;
539
540   fileHandle = SD.open(fileName);
541   if (fileHandle != 0)
542   {
543     printTraces("play %s", fileName);
544     InvAudioZero.play(fileHandle);
545     // file is closed by InvAudioZero.play()
546   }
547   else
548   {
549     printTraces("inv_play_wave : open failed
550   }
551 }
```

And now?

sensing the FUTURE

- SensorStudio 2.2 includes Lightsaber sample
- You can build your own
  - Purchase our Development Kits
  - Download SensorStudio

- Use your creativity !



Thank You