AR Gun using ESP32

Vibhore Jain, DESE, IISc

I Introduction

The following code implements AR Gun functionality on ESP32 and is tightly tied to the corresponding hardware. The code uses libraries that need to be installed before the code can compile successfuly on Arduino IDE. For uploading the code to ESP32, FT232 UART bridge with some supporting circuit was used to correctly bias the boot pins.

```
// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2
    .0)
// 6/21/2012 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
//      2019-07-08 - Added Auto Calibration and offset generator
//       - and altered FIFO retrieval sequence to avoid using blocking code
//      2016-04-18 - Eliminated a potential infinite loop
//      2013-05-08 - added seamless Fastwire support
//                 - added note about gyro calibration
//      2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
//      2012-06-20 - improved FIFO overflow handling and simplified read process
//      2012-06-19 - completely rearranged DMP initialization code and simplification
//      2012-06-13 - pull gyro and accel data from FIFO packet instead of reading directly
//      2012-06-09 - fix broken FIFO read sequence and change interrupt detection to RISING
//      2012-06-05 - add gravity-compensated initial reference frame acceleration output
//                 - add 3D math helper file to DMP6 example sketch
//                 - add Euler output and Yaw/Pitch/Roll output formats
//      2012-06-04 - remove accel offset clearing for better results (thanks Sungon Lee)
//      2012-06-01 - fixed gyro sensitivity to be 2000 deg/sec instead of 250
//      2012-05-30 - basic DMP initialization working

/* ============================================
  I2Cdev device library code is placed under the MIT license
  Copyright (c) 2012 Jeff Rowberg

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files (the "Software"), to deal
  in the Software without restriction, including without limitation the rights
  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
  copies of the Software, and to permit persons to whom the Software is
  furnished to do so, subject to the following conditions:

  The above copyright notice and this permission notice shall be included in
  all copies or substantial portions of the Software.

  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
  THE SOFTWARE.
  ===============================================
*/
#include <BleMouse.h>

BleMouse bleMouse;
#define alpha 1500
#define bit1 19
#define bit2 18
#define bit3 5
#define bit4 17
#define fire 33
#define recoil 23
#define batt_sense 34
// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
```

```
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

/* =========================================================================
   NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
   depends on the MPU-6050's INT pin being connected to the Arduino's
   external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
   digital I/O pin 2.
   ========================================================================= */

/* =========================================================================
   NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
   when using Serial.write(buf, len). The Teapot output uses this method.
   The solution requires a modification to the Arduino USBAPI.h file, which
   is fortunately simple, but annoying. This will be fixed in the next IDE
   release. For more info, see these links:

   http://arduino.cc/forum/index.php/topic,109987.0.html
   http://code.google.com/p/arduino/issues/detail?id=958
   ========================================================================= */



// uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual
// quaternion components in a [w, x, y, z] format (not best for parsing
// on a remote host such as Processing or something though)
//#define OUTPUT_READABLE_QUATERNION

// uncomment "OUTPUT_READABLE_EULER" if you want to see Euler angles
// (in degrees) calculated from the quaternions coming from the FIFO.
// Note that Euler angles suffer from gimbal lock (for more info, see
// http://en.wikipedia.org/wiki/Gimbal_lock)
//#define OUTPUT_READABLE_EULER

// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for
// more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
#define OUTPUT_READABLE_YAWPITCHROLL

// uncomment "OUTPUT_READABLE_REALACCEL" if you want to see acceleration
// components with gravity removed. This acceleration reference frame is
// not compensated for orientation, so +X is always +X according to the
// sensor, just without the effects of gravity. If you want acceleration
// compensated for orientation, us OUTPUT_READABLE_WORLDACCEL instead.
//#define OUTPUT_READABLE_REALACCEL

// uncomment "OUTPUT_READABLE_WORLDACCEL" if you want to see acceleration
// components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no magnetometer
// is present in this case). Could be quite handy in some cases.
//#define OUTPUT_READABLE_WORLDACCEL

// uncomment "OUTPUT_TEAPOT" if you want output that matches the
// format used for the InvenSense teapot demo
```

```
//#define OUTPUT_TEAPOT



#define INTERRUPT_PIN 4   // use pin 2 on Arduino Uno & most boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
unsigned long on_time;
// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
float ypr_last[3];
bool delay_flag = 0;
// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0, 0, 0, 0, 0, 0, 0, 0, 0x00, 0x00, '\r', '\n' };



// ================================================================
// ===                      INTERRUPT DETECTION ROUTINE                     ===
// ================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}



// ================================================================
// ===                      INITIAL SETUP                           ===
// ================================================================

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  pinMode(recoil, OUTPUT);
  digitalWrite(recoil,LOW);
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif
  // initialize serial communication
  // (115200 chosen because it is required for Teapot Demo output, but it's
  // really up to you depending on your project)
  Serial.begin(115200);
  Serial.println("Starting BLE work!");
  bleMouse.begin();
  while (!bleMouse.isConnected());
  delay(2000);
  while (!Serial); // wait for Leonardo enumeration, others continue immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or Arduino
  // Pro Mini running at 3.3V, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.
```

```
201
202   // initialize device
203   Serial.println(F("Initializing I2C devices..."));
204   mpu.initialize();
205   pinMode(INTERRUPT_PIN, INPUT);
206
207   // verify connection
208   Serial.println(F("Testing device connections..."));
209   Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection
        failed"));
210
211   // wait for ready
212   Serial.println(F("\nSend any character to begin DMP programming and demo: "));
213   //while (Serial.available() && Serial.read()); // empty buffer
214   //while (!Serial.available());                 // wait for data
215   //while (Serial.available() && Serial.read()); // empty buffer again
216
217   // load and configure the DMP
218   Serial.println(F("Initializing DMP..."));
219   devStatus = mpu.dmpInitialize();
220
221   // supply your own gyro offsets here, scaled for min sensitivity
222   mpu.setXGyroOffset(220);
223   mpu.setYGyroOffset(76);
224   mpu.setZGyroOffset(-85);
225   mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
226
227   // make sure it worked (returns 0 if so)
228   if (devStatus == 0) {
229     // Calibration Time: generate offsets and calibrate our MPU6050
230     mpu.CalibrateAccel(6);
231     mpu.CalibrateGyro(6);
232     mpu.PrintActiveOffsets();
233     // turn on the DMP, now that it's ready
234     Serial.println(F("Enabling DMP..."));
235     mpu.setDMPEnabled(true);
236
237     // enable Arduino interrupt detection
238     Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
239     Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
240     Serial.println(F(")..."));
241     attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
242     mpuIntStatus = mpu.getIntStatus();
243
244     // set our DMP Ready flag so the main loop() function knows it's okay to use it
245     Serial.println(F("DMP ready! Waiting for first interrupt..."));
246     dmpReady = true;
247
248     // get expected DMP packet size for later comparison
249     packetSize = mpu.dmpGetFIFOPacketSize();
250   } else {
251     // ERROR!
252     // 1 = initial memory load failed
253     // 2 = DMP configuration updates failed
254     // (if it's going to break, usually the code will be 1)
255     Serial.print(F("DMP Initialization failed (code "));
256     Serial.print(devStatus);
257     Serial.println(F(")"));
258   }
259
260   // configure LED for output
261
262   pinMode(fire , INPUT_PULLUP);
263   pinMode(bit1 , INPUT_PULLUP);
264   pinMode(bit2 , INPUT_PULLUP);
265   pinMode(bit3 , INPUT_PULLUP);
266   pinMode(bit4 , INPUT_PULLUP);
267 }
268
269
270
```

```
// ================================================================
// ===                      MAIN PROGRAM LOOP                     ===
// ================================================================

void loop() {
  // if programming failed, don't try to do anything
  if (!dmpReady) return;
  // blink LED to indicate activity
  blinkState = !blinkState;
  digitalWrite(LED_PIN, blinkState);
  if (bleMouse.isConnected()) {
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
#ifdef OUTPUT_READABLE_YAWPITCHROLL
      mpu.dmpGetQuaternion(&q, fifoBuffer);
      mpu.dmpGetGravity(&gravity, &q);
      mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
#endif


      // display Euler angles in degrees

      /*Serial.print("ypr\t");
      Serial.print(ypr[0] * 180 / M_PI);
      Serial.print("\t");
      Serial.print(ypr[1] * 180 / M_PI);
      Serial.print("\t");
      Serial.println(ypr[2] * 180 / M_PI);
*/
      if(digitalRead(bit1)==HIGH)
        bleMouse.move(int(alpha*ypr[0] - alpha*ypr_last[0]), int(alpha*ypr[2] - alpha*ypr_last[2]),
    0);
      //Serial.print(int(ypr[0] - ypr_last[0]));
      //Serial.print("\t");
      //Serial.println(int(ypr[2] - ypr_last[2]));
    }
    if(int(alpha*ypr[0] - alpha*ypr_last[0]) != 0)
      ypr_last[0] = ypr[0];
    if(int(alpha*ypr[2] - alpha*ypr_last[2]) !=0)
      ypr_last[2] = ypr[2];
    if(digitalRead(fire)==LOW)
      {
        bleMouse.press(MOUSE_LEFT);
        on_time = millis();
        digitalWrite(recoil,HIGH);
      }
    else
      bleMouse.release(MOUSE_LEFT);
    if(digitalRead(bit4) == LOW)
      bleMouse.move(0,0,1);
    if(digitalRead(bit3) == LOW)
      bleMouse.press(MOUSE_RIGHT);
    else
      bleMouse.release(MOUSE_RIGHT);
    if(digitalRead(bit2) == LOW)
      bleMouse.press(MOUSE_MIDDLE);
    else
      bleMouse.release(MOUSE_MIDDLE);
    if(millis()-on_time > 40)
      digitalWrite(recoil,LOW);
  }
}
```

II Schematic

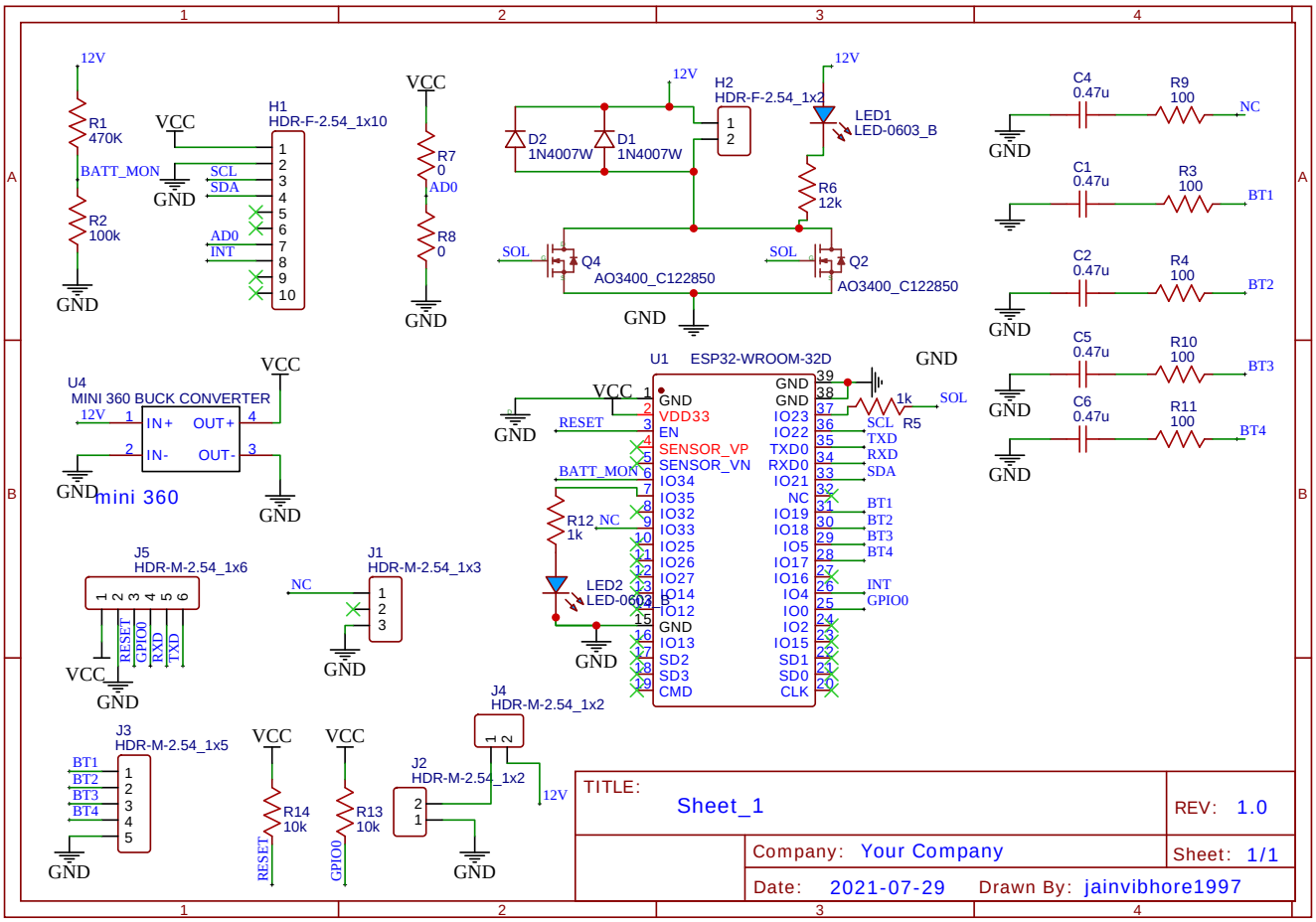

Figure 1: Schematic of augmented reality gun