Lampiran: Algoritma GPS Adafruit modul MTK3329

```
// Test code for Adafruit GPS modules using MTK3329/MTK3339 driver
//
// This code shows how to listen to the GPS module in an interrupt
// which allows the program to have more 'freedom' - just parse
// when a new NMEA sentence is available! Then access data when
// desired.
//
// Tested and works great with the Adafruit Ultimate GPS module
// using MTK33x9 chipset
//    ------> http://www.adafruit.com/products/746
// Pick one up today at the Adafruit electronics shop
// and help support open source hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>

#define GPSSerial Serial1

// If you're using a GPS module:
// Connect the GPS Power pin to 5V
// Connect the GPS Ground pin to ground
// If using software serial (sketch example default):
//   Connect the GPS TX (transmit) pin to Digital 3
//   Connect the GPS RX (receive) pin to Digital 2
// If using hardware serial (e.g. Arduino Mega):
//   Connect the GPS TX (transmit) pin to Arduino RX1, RX2 or RX3
//   Connect the GPS RX (receive) pin to matching TX1, TX2 or TX3

// If you're using the Adafruit GPS shield, change
// SoftwareSerial mySerial(3, 2); -> SoftwareSerial mySerial(8, 7);
// and make sure the switch is set to SoftSerial

// If using software serial, keep this line enabled
// (you can change the pin numbers to match your wiring):
//HardwareSerial mySerial = Serial1;
//SoftwareSerial mySerial(19,18);

// If using hardware serial (e.g. Arduino Mega), comment out the
// above SoftwareSerial line, and enable this line instead
// (you can change the Serial number to match your wiring):
Adafruit_GPS GPS(&GPSSerial);
//HardwareSerial mySerial = Serial1;


//Adafruit_GPS GPS(&mySerial);


// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences.
```

```
#define GPSECHO  true

// this keeps track of whether we're using the interrupt
// off by default!
boolean usingInterrupt = false;
void useInterrupt(boolean); // Func prototype keeps Arduino 0023 happy

void setup()
{

  // connect at 115200 so we can read the GPS fast enough and echo without dropping chars
  // also spit it out
  Serial.begin(9600);
  Serial.println("Adafruit GPS library basic test!");


  // 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800

  GPS.begin(9600);


  // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including
altitude

  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

  // uncomment this line to turn on only the "minimum recommended" data

  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);

  // For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since

  // the parser doesn't care about other sentences at this time


  // Set the update rate

  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz update rate

  // For the parsing code to work nicely and have time to sort thru the data, and

  // print it out we don't suggest using anything higher than 1 Hz


  // Request updates on antenna status, comment out to keep quiet

  GPS.sendCommand(PGCMD_ANTENNA);


  // the nice thing about this code is you can have a timer0 interrupt go off

  // every 1 millisecond, and read data from the GPS for you. that makes the

  // loop code a heck of a lot easier!

  useInterrupt(true);
```

```
  delay(1000);
  // Ask for firmware version
 // mySerial.println(PMTK_Q_RELEASE);
}


// Interrupt is called once a millisecond, looks for any new GPS data, and stores it
SIGNAL(TIMER0_COMPA_vect) {
  char c = GPS.read();
  // if you want to debug, this is a good time to do it!
#ifdef UDR0
  if (GPSECHO)
    if (c) UDR0 = c;
    // writing direct to UDR0 is much much faster than Serial.print
    // but only one character can be written at a time.
#endif
}

void useInterrupt(boolean v) {
  if (v) {
    // Timer0 is already used for millis() - we'll just interrupt somewhere
    // in the middle and call the "Compare A" function above
    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    // do not call the interrupt function COMPA anymore
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}
```

```
uint32_t timer = millis();
void loop()                     // run over and over again
{
  // in case you are not using the interrupt above, you'll
  // need to 'hand query' the GPS, not suggested :(
  if (! usingInterrupt) {
    // read data from the GPS in the 'main loop'
    char c = GPS.read();
    // if you want to debug, this is a good time to do it!
    if (GPSECHO)
      if (c) Serial.print(c);
  }

  // if a sentence is received, we can check the checksum, parse it...
  if (GPS.newNMEAreceived()) {
    // a tricky thing here is if we print the NMEA sentence, or data
    // we end up not listening and catching other sentences!
    // so be very wary if using OUTPUT_ALLDATA and trytng to print out data
    //Serial.println(GPS.lastNMEA());   // this also sets the newNMEAreceived() flag to false

    if (!GPS.parse(GPS.lastNMEA()))   // this also sets the newNMEAreceived() flag to false
      return;  // we can fail to parse a sentence in which case we should just wait for another
  }

  // if millis() or timer wraps around, we'll just reset it
  if (timer > millis())  timer = millis();

  // approximately every 2 seconds or so, print out the current stats
  if (millis() - timer > 2000) {
    timer = millis(); // reset the timer
```

```
    Serial.print("\nTime: ");
    Serial.print(GPS.hour, DEC); Serial.print(':');
    Serial.print(GPS.minute, DEC); Serial.print(':');
    Serial.print(GPS.seconds, DEC); Serial.print('.');
    Serial.println(GPS.milliseconds);
    Serial.print("Date: ");
    Serial.print(GPS.day, DEC); Serial.print('/');
    Serial.print(GPS.month, DEC); Serial.print("/20");
    Serial.println(GPS.year, DEC);
    Serial.print("Fix: "); Serial.print((int)GPS.fix);
    Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
    if (GPS.fix) {
      Serial.print("Location: ");
      Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);
      Serial.print(", ");
      Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);
      Serial.print("Location (in degrees, works with Google Maps): ");
      Serial.print(GPS.latitudeDegrees, 4);
      Serial.print(", ");
      Serial.println(GPS.longitudeDegrees, 4);

      Serial.print("Speed (knots): "); Serial.println(GPS.speed);
      Serial.print("Angle: "); Serial.println(GPS.angle);
      Serial.print("Altitude: "); Serial.println(GPS.altitude);
      Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
    }
  }
}
```