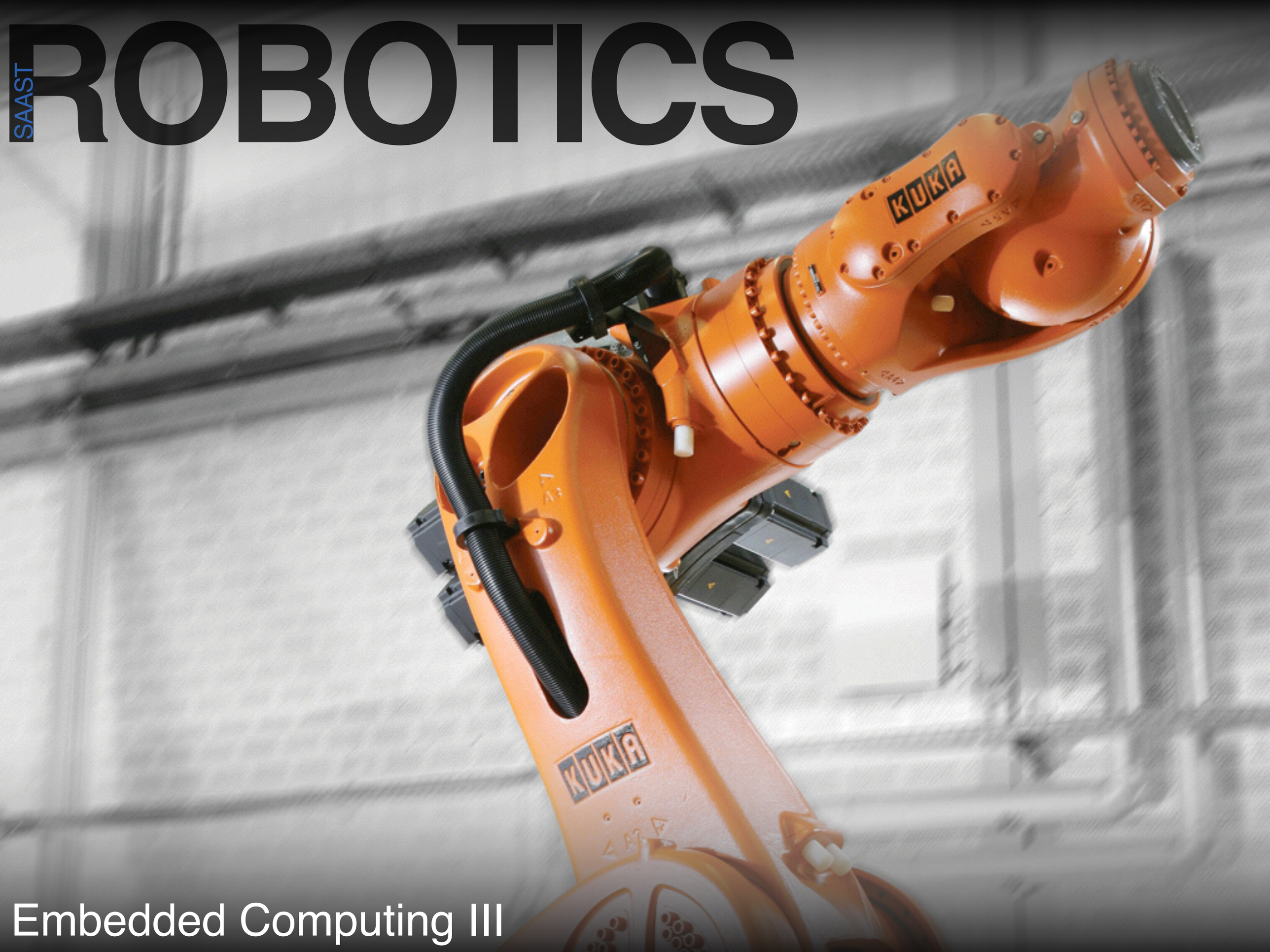
SAAST

ROBOTICS



Embedded Computing III

variable type modifiers

to preserve the value of a variable between successive subroutine calls

`static`

```
ISR(TIMER3_COMPA_vect)
{
  static long L_encoder_last=0, R_encoder_last=0;
  int L_velocity_raw, R_velocity_raw;

  // calcuate velocity in 10 * ticks (current - previous) per 0.01 sec

  L_velocity_raw = -EGAIN*(L_encoder - L_encoder_last);
  L_encoder_last = L_encoder;
  L_velocity = (float)L_velocity*V_FILTER + (1-V_FILTER)*(float)L_velocity_raw;

  R_velocity_raw = -EGAIN*(R_encoder - R_encoder_last);
  R_encoder_last = R_encoder;
  R_velocity = (float)R_velocity*V_FILTER + (1-V_FILTER)*(float)R_velocity_raw;
}
```

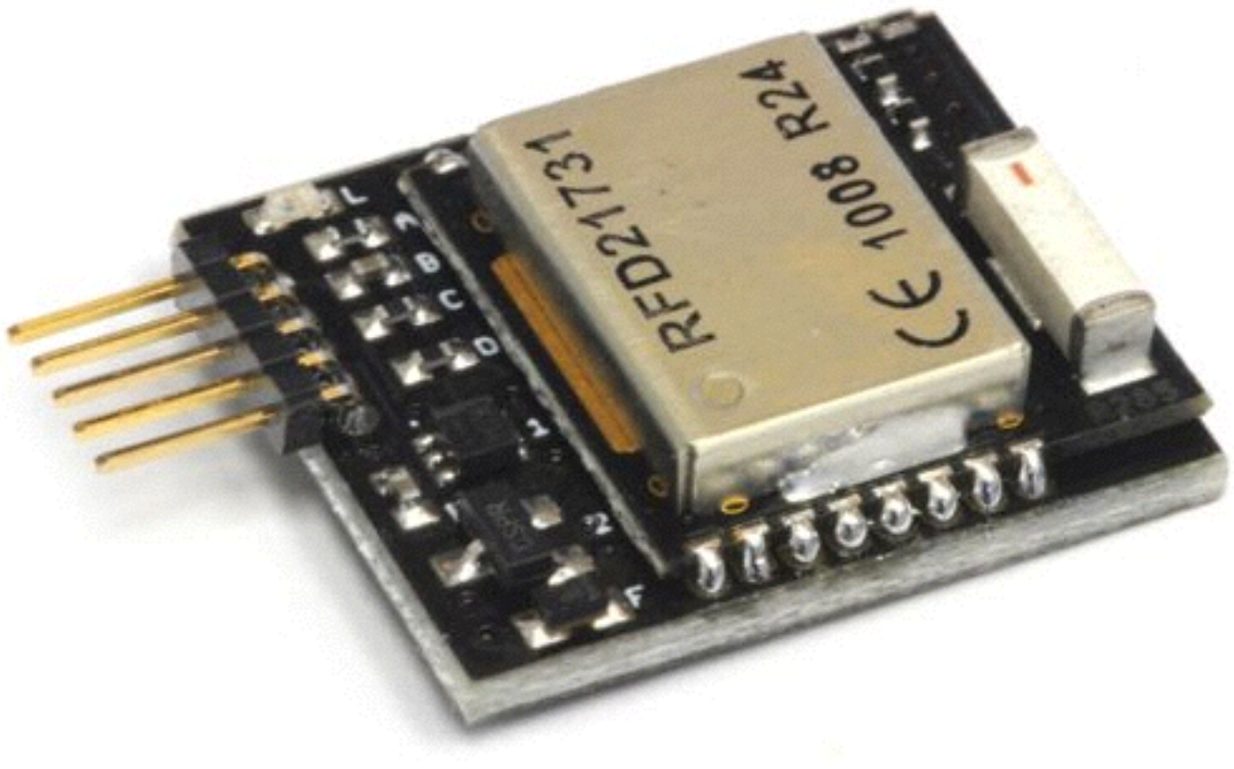
variable type modifiers

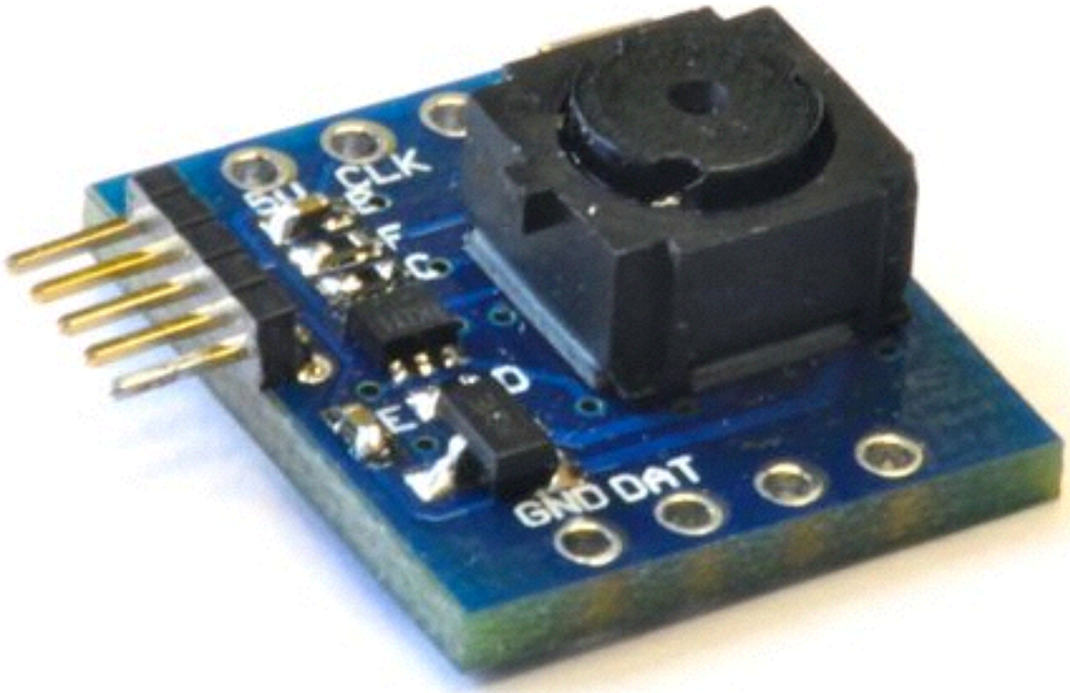
to alert the compiler that a variable may change outside the routine

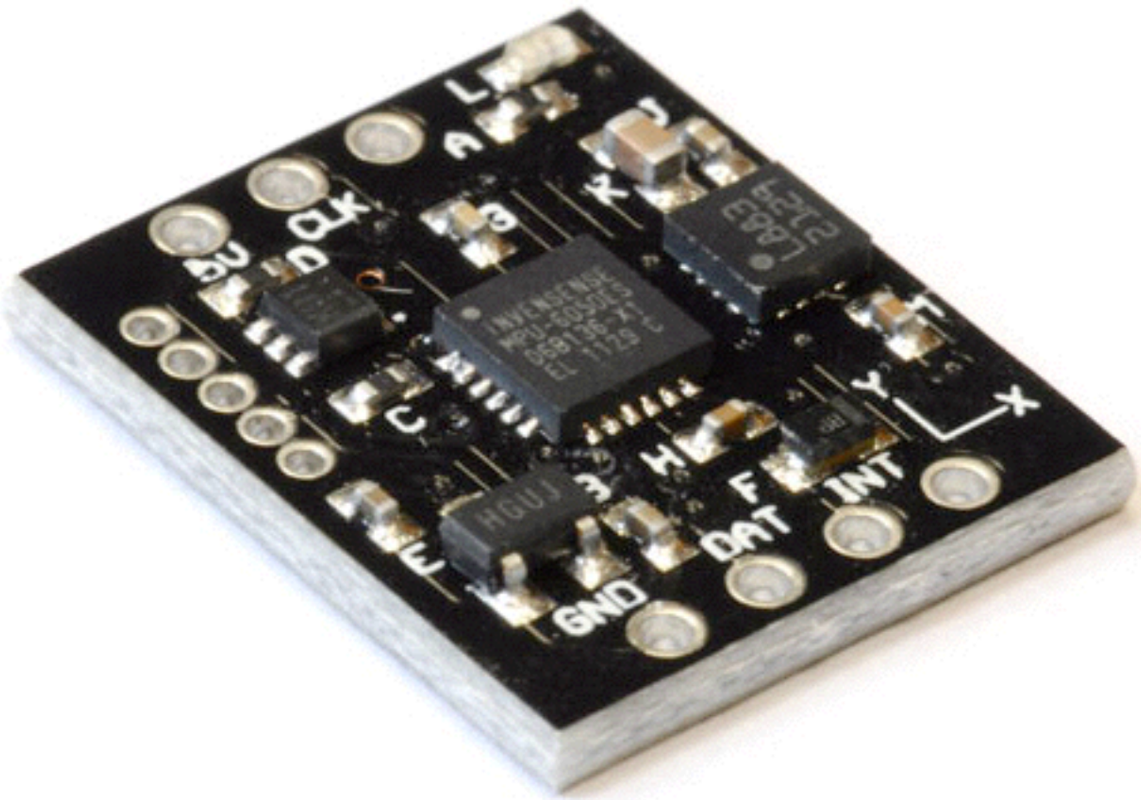
volatile

```
volatile int flag = 0;
char message[3] = {0, 0, 0};

int main(void){
  while(1){
    if(flag){
      toggle(PORTE,6); // toggle the green LED
      flag = 0;
    }
  }
}

ISR(PCINT0_vect)
{
  if(!check(PINB,5))
  {
     flag = 1;
     RFreceive(message);
  }
}
```

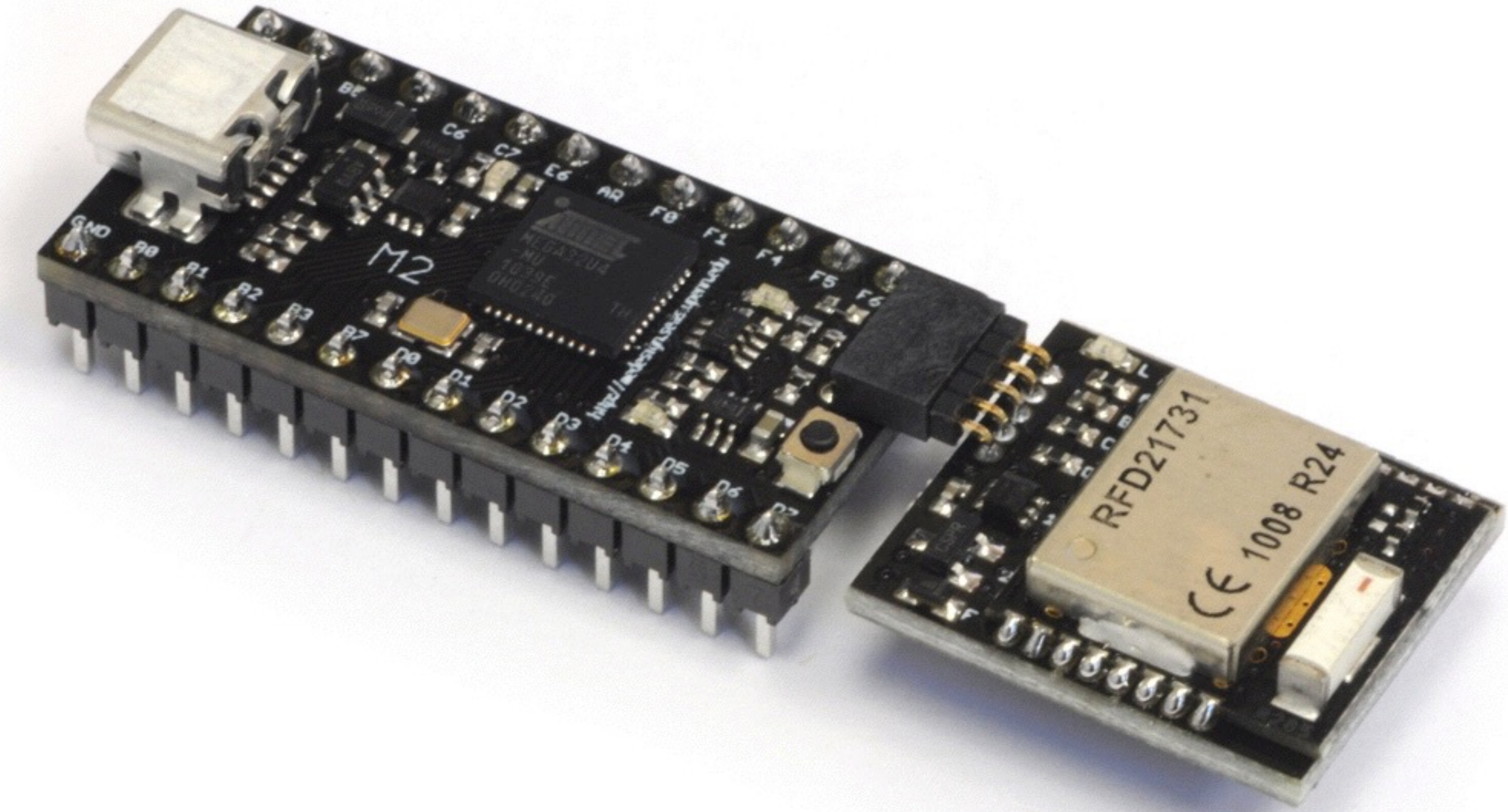
Peripherals







mRF Wireless Module



mRF functions

`m_rf_open(channel, RXaddress, packet_length);`

listen to a desired `RXaddress` (0x00 to 0xFF)
over a wireless `channel` (1-32, TX/RX must match)
with a specified `packet_length` (1-32, TX/RX must match).

`m_rf_read(buffer, packet_length);`

extract `packet_length` (1-32, TX/RX must match) bytes
into a local array `buffer`

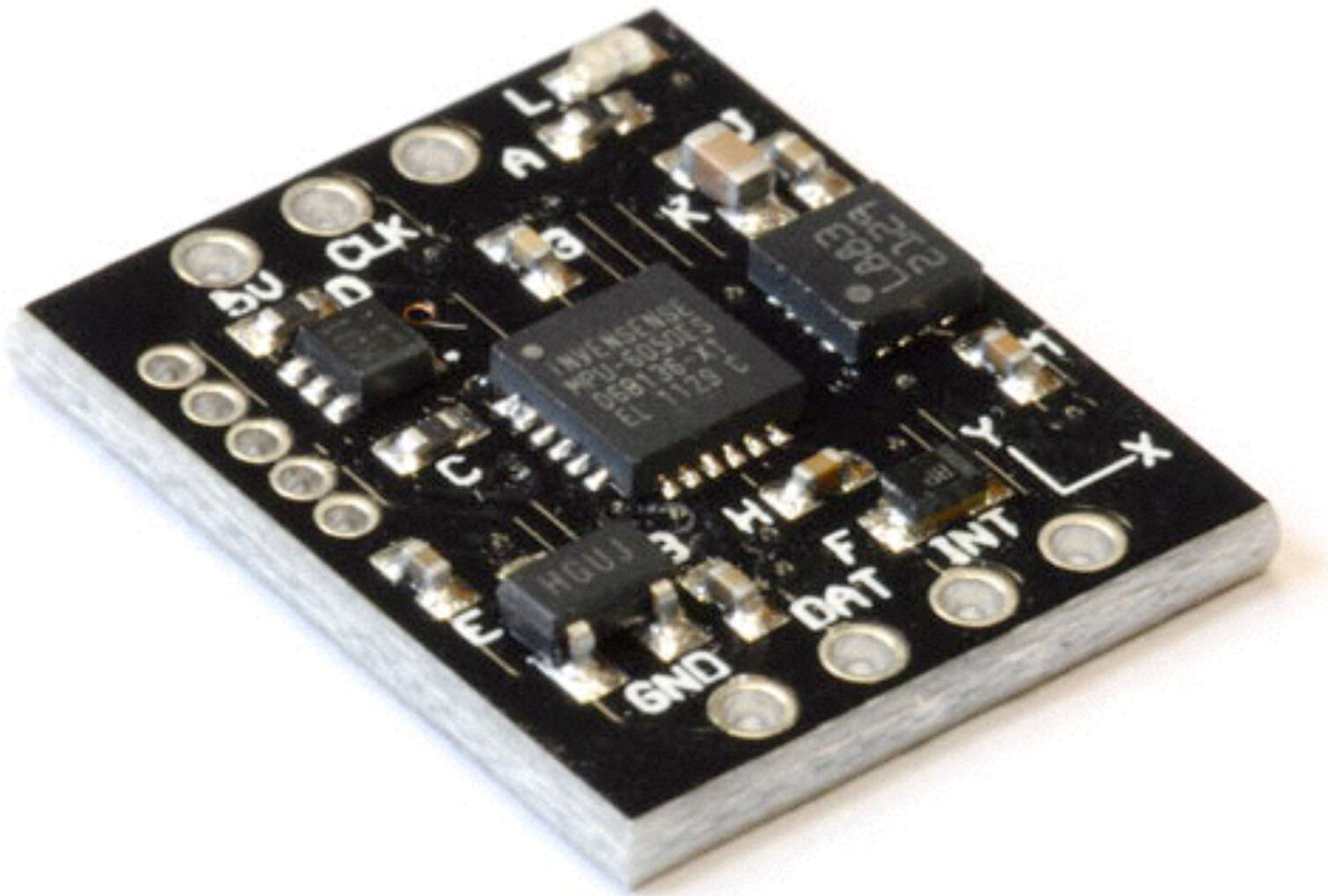
`m_rf_send(TXaddress, buffer, packet_length);`

send `packet_length` (1-32, TX/RX must match) bytes
stored in a local array `buffer`
to a desired `TXaddress` (0x00 to 0xFF)

Wireless

```
#include "saast.h"

#define RF_CHANNEL 12
#define RF_LENGTH  4

char incoming[RF_LENGTH] = {0,0,0,0};        // incoming wireless buffer
char outgoing[RF_LENGTH] = {0,0,0,0};        // outgoing wireless buffer
char me = 0xAB, you = 0xCD;                  // wireless addresses

int main(void)
{
  m_init();                                  // initialize the m2
  m_rf_open(RF_CHANNEL,me,RF_LENGTH);        // open the wireless channel

  while(1){
   m_wait(200);                              // delay 200 ms
   if(incoming[0] == 0x12){
   m_rf_send(you,outgoing,RF_LENGTH));       // send a wireless packet
   }
  }
}

// interrupt handler to process incoming mRF packets
ISR(INT2_vect){
  m_rf_read(incoming,RF_LENGTH);
}
```

mIMU 9-DOF inertial sensor



mIMU functions

`m_imu_init(accel_scale, gyro_scale);`

initialize the m2 to talk to the mIMU.
with `accel_scale` in +/-g as 0=2, 1=4, 2=8, 3=16
and `gyro_scale` in +/-deg/s as 0=250, 1=500, 2=1000, 3=2000

`m_imu_raw(buffer);`

extract 9 elements into local **integer** array `buffer`

`buffer` will consist of three three-byte packets corresponding to the three sensors (Accelerometer, Rate Gyro, Magnetometer):

0	1	2	3	4	5	6	7	8
Ax	Ay	Az	Gx	Gy	Gz	Mx	My	Mz

```c
#include "saast.h"

#define ACCEL_SCALE 2
#define GYRO_SCALE  1

int main(void)
{

  int raw_imu_buffer[9];
  m_init();                                  // initialize the m2
  m_imu_init(ACCEL_SCALE,GYRO_SCALE);        // initialize the mIMU

   while(1){
    m_wait(50);                              // delay 50 ms
    m_imu_raw(raw_imu_buffer);               // get mIMU data
    if(raw_imu_buffer[0] > 120){             // check Ax
      m_green(ON);
    } else {
      m_green(OFF);
    }
   }
}
```