List of functions of EMoRo 2560 library (Arduino)

Function prototype	Function description	Page
`int InitEmoro(void);`	Initialize controller	4
`int ReadEmoroHardware(void);`	Reading an initialized hardware	4
`int EmoroServo.attach(unsigned char port);`	Initialize servo motor	6
`int EmoroServo.detach(unsigned char port);`	Releasing resources of servo motor	6
`int EmoroServo.write(unsigned char port, int position);`	Setting the position of the servo motor	7
`int EmoroServo.read(unsigned char port);`	Reading the position of the servomotor	8
`float ReadPowerSupply(void);`	Reading the power supply voltage	10
`int Ultrasonic.attach(unsigned char port);`	Initialize ultrasonic sensor	11
`int Ultrasonic.detach(unsigned char port);`	Releasing resources of ultrasonic sensor	11
`int Ultrasonic.read(unsigned char port);`	Reading the ultrasonic sensor	12
`int Lcd.init(void);`	Initialize LCD 2x16	14
`int Lcd.clear(void);`	Function clears LCD 2x16	14
`int Lcd.printString(char *str);`	Function prints a string on LCD 2x16	15
`int Lcd.printChar(char data);`	Function prints a character on LCD 2x16	15

`long Lcd.print(val);`
`long Lcd.print(val, format);`
	Function prints the ASCII text on the LCD 2x16	16

`long Lcd.write(val);`
`long Lcd.write(str);`
`long Lcd.write(buf, len);`
	Function sends a byte or set of bytes on the LCD 2x16	17

`int Lcd.locate(unsigned char r, unsigned char c);`	Function sets the print location of LCD	18
`int Lcd.contrast(unsigned char con);`	Function adjusts the LCD contrast	18
`void Lcd.backlightOn(void);`	Function turns On the LCD backlight	19
`void Lcd.backlightOff(void);`	Function turns Off the LCD backlight	20
`int Acc.init(void);`	Initialize accelerometer	21
`unsigned char Acc.testConnection(void);`	Function verify accelerometer connection	21
`int Acc.read(int *x, int *y, int *z);`	Reads 3-axis accelerometer (x, y, z)	22

`int Acc.readX(void);`
`int Acc.readY(void);`
`int Acc.readZ(void);`
	Function reads the acceleration axis X, Y or Z	23

`int Gyr.initBasic(void);`	Initialize gyroscope in Basic mode	25
`int Gyr.initBasic(unsigned int odr, unsigned int range);`	Initialize gyroscope in Basic mode with specified parameters	25
`int Gyr.init(void);`	Initialize gyroscope in Advanced mode (with automatic calculation of rotation angle)	26
`int Gyr.init(unsigned int odr, unsigned int range);`	Initialize gyroscope in Advanced mode with specified parameters	27
`int Gyr.stop(void);`	Stops the Advanced mode of gyroscope	28
`unsigned char Gyr.testConnection(void);`	Function verify gyroscope connection	29
`int Gyr.readDegrees(double *x, double *y, double *z);`	Reading the rotation angle in gyros Advanced mode[rotation angle in degrees]	30

`double Gyr.readDegreesX(void);`
`double Gyr.readDegreesY(void);`
`double Gyr.readDegreesZ(void);`
	Reading the rotation angle in X, Y or Z axis in gyros Advanced mode[rotation angle in degrees]	31

`int Gyr.setDegrees(double x, double y, double z);`	Setting the current position of the gyroscope in Advanced mode as the specified value	32
`int Gyr.resetDegrees(void);`	Setting the current position of the gyroscope in Advanced mode to 0 degrees for all three axes	34
`int Gyr.read(int *x, int *y, int *z);`	Function reads the gyroscopes axis (x, y, z) [angular acceleration]	35

`int Gyr.readX(void);`
`int Gyr.readY(void);`
`int Gyr.readZ(void);`
	Function reads the angular acceleration axis X, Y or Z.	36

`unsigned long Gyr.readCounterX(void);`
`unsigned long Gyr.readCounterY(void);`
`unsigned long Gyr.readCounterZ(void);`
	Function returns the number of sensor readings that have passed the filter in the Advanced mode for X, Y or Z axis	37

`unsigned char Gyr.Dready(void);`	Returns state of gyroscopes DRDY signal	38
`int Gyr.Calibration(unsigned int samples);`	Gyroscope calibration with a specified sample numbers	38

`int Gyr.dc_offsetX(void);`
`int Gyr.dc_offsetY(void);`
`int Gyr.dc_offsetZ(void);`
	Returns a value of the calculated DC offsets for X, Y or Z axis, which is given during calibration	40

`int Gyr.noiseX (void);`
`int Gyr.noiseY (void);`
`int Gyr.noiseZ (void);`
	Returns a value of the calculated noise for X, Y or Z axis, which is given during calibration	40

`int Mag.init(void);`	Initialize compass	41
`unsigned char Mag.testConnection(void);`	Function verify compass connection	41
`int Mag.read(int *direction, int *inclination,`	Function reads the direction, inclination	42

`int *strength);`	and strenght of the magnetic field	
`int Mag.readDirection(void);`	Function reads the compass direction	43
`int Mag.readInclination(void);`	Function reads the compass inclination	44
`int Mag.readStrenght(void);`	Reads the strenght of the magnetic field	46
`int Mag.calibrate(unsigned char step);`	Function calibrates the compass	47
`void InitSwitch(void);`	Initialize pushbuttons SW_1-SW_4	48
`unsigned char ReadSwitch(unsigned char sw);`	Reads the state of pushbuttons SW_1-SW_4	48
`int Bluetooth.init(void);`	Initialize bluetooth module	50
`unsigned char Bluetooth.testConnection(void);`	Function verify bluetooth connection	50
`int Bluetooth.changeName(char *new_name);`	Changes name of the bluetooth module	51
`int Bluetooth.changePasskey(char *new_passkey);`	Changes passkey of the bluetooth module	52
`int Bluetooth.changeNameAndPasskey(char *new_name,` `char *new_passkey);`	Changes name and passkey of the bluetooth module	53
`char *Bluetooth.readName(void);`	Reads name of bluetooth module	54
`char *Bluetooth.readPasskey(void);`	Reads passkey of bluetooth module	55
`unsigned char Bluetooth.connection(void);`	Reads status of the bluetooth connection	55
`int DS1820.attach(unsigned char port);`	Initialize temperature sensor DS1820	57
`float DS1820.read(unsigned char port);`	Reads temperature sensor DS1820	57

```
int InitEmoro(void);
```

Function initializes the main settings of the EMoRo 2560 controller. In Arduino IDE function is called before the **setup();** and **loop();** functions and its re-call is not required. In the case that the "EMoRo extend board - EMoRo GLAM" is connected to the EMoRo 2560 controller, function will initialize the available modules: Bluetooth, compass, gyroscope, accelerometer, LCD and pushbuttons (SW_1 - SW_4). If pushbutton "SW1" is activated, bluetooth module will be set to default settings. (Name: EMoRo 2560, Passkey:0000). If pushbutton SW_2 on EMoRo GLAM module is activated, function for calibration of the compass will start up.

Function prototype:

emoro_2560.h

Arguments:

None

Return value:

```
int   –  Result of function:
                  (0)   –  Initialized EMoRo 2560 controller
          (bxxxxxxx1)   –  Initialized EMoRo GLAM board – LCD avilable
          (bxxxxxx1x)   –  Initialized EMoRo GLAM board – Pushbuttons available
          (bxxxxx1xx)   –  Initialized EMoRo GLAM board – Bluetooth available
          (bxxxx1xxx)   –  Initialized EMoRo GLAM board – Accelerometer available
          (bxxx1xxxx)   –  Initialized EMoRo GLAM board – Gyroscope available
          (bxx1xxxxx)   –  Initialized EMoRo GLAM board – Compass available
```

Supported moduls:

EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                              /* Arduino setup                                  */

   /* InitEmoro()is executed and re-call is not required                                     */

   /* Pushbutton SW_1 set default settings for the Bluetooth module.                       */
   /* Pushbutton SW_2 runs a program for the calibration of the compass.                 */

   Serial.begin(9600);                         /* initialize UART 9600 bps                     */
   Serial.println("Example: InitEmoro();");    /* send Examples name                              */

   if(ReadEmoroHardware() & LCD_AVAILABLE)     /* if LCD on EMoRo GLAM is initialized      */
      Lcd.print("InitEmoro();");               /* print examples name                             */
}

void loop(void){                               /* Arduino loop                                    */
}
```

```
int ReadEmoroHardware(void);
```

Function reads initialized hardware of the EMoRo 2560 controller. Before calling the setup (); and loop (); functions, EMoRo controller will initialize the available controllers hardware and ReadEmoroHardware (); function can check availability of EMoRo GLAM module, gyroscope, compass, accelerometer, Bluetooth, LCD and pushbuttons SW_1 - SW_4;

Function prototype:

emoro_2560.h

Arguments:

None

Masks of function results:

```
      LCD_AVAILABLE            (0x01)
      SW_AVAILABLE             (0x02)
```

```
BLUETOOTH_AVAILABLE      (0x04)
ACC_AVAILABLE            (0x08)
GYR_AVAILABLE            (0x10)
MAG_AVAILABLE            (0x20)
```

Return value:

```
int  –  Result of function:
                 (0)   –  Initialized EMoRo 2560 controller
        (bxxxxxxx1)   –  Initialized EMoRo GLAM board – LCD avilable
        (bxxxxxx1x)   –  Initialized EMoRo GLAM board – Pushbuttons available
        (bxxxxx1xx)   –  Initialized EMoRo GLAM board – Bluetooth available
        (bxxxx1xxx)   –  Initialized EMoRo GLAM board – Accelerometer available
        (bxxx1xxxx)   –  Initialized EMoRo GLAM board – Gyroscope available
        (bxx1xxxxx)   –  Initialized EMoRo GLAM board – Compass available
```

Supported moduls:

EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                /* Arduino setup                                */

   Serial.begin(9600);                                   /* initialize UART 9600 bps                */
   Serial.println("Example: ReadEmoroHardware();");      /* send Examples name                  */

   Serial.print("EMoRo HW = ");                 /* print initialized hardware                  */
   Serial.println(ReadEmoroHardware(), BIN);    /* print HW as BIN number                       */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized      */
      Lcd.locate(0, 0);                         /* print initialized hardware                  */
      Lcd.print("EMoRo HW =");
      Lcd.locate(1, 0);
      Lcd.print(ReadEmoroHardware(), BIN);      /* print HW as BIN number                       */
   }
}

void loop(void){                                 /* Arduino loop                                 */
}
```

```
int EmoroServo.attach(unsigned char port);
```

Function initialize output of servo motor on ports SERVO_0 – SERVO_7. After initialization of servo motor default state of output is set to value 1500 which corresponds to servo impulse of 1.5ms. Initialized driver of servo motor generates servo impulses in the interval [0,5ms - 2.5ms] every 20ms.

Function prototype:
    emoro_servo.h

Arguments:
    unsigned char port – Servo [SERVO_0 – SERVO_7]

Return value:
```
     int   –  Result of function:
              (0)   –  Servo output successfully initialized
              (-1)  –  Error: servo port out of range [SERVO_0 – SERVO_7]
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                           /* Arduino setup                                        */

   Serial.begin(9600);                                /* initialize UART 9600 bps                  */
   Serial.println("Example: EmoroServo.attach();");   /*    send Examples name                     */

   EmoroServo.attach(SERVO_0);              /* initialize servo motor on port SERVO_0             */

   EmoroServo.write(SERVO_0, 1000);         /* set SERVO_0 pulse 1ms                            */

   if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized      */
      Lcd.print("Servo.attach();");         /* print examples function                            */
      Lcd.locate(1, 0);                     /* set LCD print location to 1, 0               */
      Lcd.print("SERVO_0 = 1 ms");          /* print SERVO_0 output                            */
   }
}

void loop(void){                            /* Arduino loop                                         */
}
```

```
int EmoroServo.detach(unsigned char port);
```

Function releases resources of the servo motor on ports SERVO_0 – SERVO_7. Before releasing of resources servo motor port must be initialized with **EmoroServo.attach();** function.

Function prototype:
    emoro_servo.h

Arguments:
    unsigned char port – Servo [SERVO_0 – SERVO_7]

Return value:
```
     int   –  Result of function:
              (0)   –  Servo motor resources successfully released
              (-1)  –  Error: Argument "port" is out of range [SERVO_0 – SERVO_7]
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                               /* Arduino setup                                        */

   Serial.begin(9600);                                     /* initialize UART 9600 bps             */
   Serial.println("Example: EmoroServo.detach();");     /* send Examples name                   */

   if(ReadEmoroHardware() & LCD_AVAILABLE)  /* if LCD on EMoRo GLAM is initialized      */
      Lcd.print("Servo.detach();");          /* print Examples function                            */
}

void loop(void){                                /* Arduino loop                                         */
   if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized      */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0               */
      Lcd.print("attach();");                   /* print status of SERVO_1 output                         */
   }
   Serial.println("attach();");                 /* send state of SERVO_0 output                         */
   EmoroServo.attach(SERVO_0);                  /* initialize servo motor on port SERVO_0            */
   EmoroServo.write(SERVO_0, 1000);             /* set SERVO_0 pulse 1ms                           */
   delay(2000);                                 /* wait 2000 ms                                         */

   if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized      */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0                  */
      Lcd.print("detach();");                   /* print state of SERVO_0 output                        */
   }
   Serial.println("detach();");                 /* send state of SERVO_0 output                         */
   EmoroServo.detach(SERVO_0);                  /* release resources of servo motor from SERVO_0 port    */
   delay(2000);                                 /* wait 2000 ms                                         */
}
```

```
int EmoroServo.write(unsigned char port, int position);
```

Function sets a new position of servo motor in range [500 – 1500 us] on ports SERVO_0 – SERVO_7. Before setting a new position of servo motor it is necessary to initialize servo motor with **EmoroServo.attach();** function.

Function prototype:

    emoro_servo.h

Arguments:

    unsigned char port - Servo [SERVO_0 – SERVO_7]
    int position   - Position of servo motor [500 - 2500] us

Return value:

    int – Result of function:
        (0) – Servo output successfully set to a new value
        (-1) – Error: Position is out of range [500-2500]
        (-2) – Error: Argument "port" is out of range [SERVO_0 – SERVO_7]
        (-3) – Error: Servo motor is not initialized

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                               /* Arduino setup                                        */

   Serial.begin(9600);                                     /* initialize UART 9600 bps             */
   Serial.println("Example: EmoroServo.write();");      /* send Examples name                   */

   EmoroServo.attach(SERVO_0);                  /* initialize servo motor on port SERVO_0            */
   if(ReadEmoroHardware() & LCD_AVAILABLE)  /* if LCD on EMoRo GLAM is initialized     */
      Lcd.print("Servo.write();");          /* print Examples function                   */
}

void loop(void){                            /* Arduino loop                               */

   if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized     */
      Lcd.locate(1, 0);                     /* set LCD print location to 1, 0          */
      Lcd.print("write(1000);");            /* print position of SERVO_0 output           */
   }
   Serial.println("write(1000);");          /* send position of SERVO_0 output            */
   EmoroServo.write(SERVO_0, 1000);         /* set SERVO_0 pulse 1ms                   */
   delay(2000);                             /* wait 2000 ms                               */

   if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized     */
      Lcd.locate(1, 0);                     /* set LCD print location to 1, 0          */
      Lcd.print("write(2000);");            /* print position of SERVO_0 output           */
   }
   Serial.println("write(2000);");          /* send position of SERVO_0 output            */
   EmoroServo.write(SERVO_0, 2000);         /* set SERVO_0 pulse 2ms                   */
   delay(2000);                             /* wait 2000 ms                               */
}
```

```
int EmoroServo.read(unsigned char port);
```

Function reads the position of servo motor in range [500 – 1500 us] on ports SERVO_0 – SERVO_7. Before reading the position of servo motor it is necessary to initialize servo motor with **EmoroServo.attach();** function.

Function prototype:
```
     emoro_servo.h
```

Arguments:
```
     unsigned char port -  Servo [SERVO_0 - SERVO_7]
```

Return value:
```
     int   - Result of function:
               (500-2500) - Position of servo motor
               (-1)       - Error: Argument "port" is out of range [SERVO_0 - SERVO_7]
               (-2)       - Error: Servo motor is not initialized
```

Supported moduls:
```
     EMoRo 2560, Extend board - EMoRo GLAM
```

Example:
```
void setup(void){                           /* Arduino setup                              */

   Serial.begin(9600);                             /* initialize UART 9600 bps          */
   Serial.println("Example: EmoroServo.read();");  /* send Examples name                */

   EmoroServo.attach(SERVO_0);              /* initialize servo motor on port SERVO_0        */
   EmoroServo.attach(SERVO_1);              /* initialize servo motor on port SERVO_1        */

   EmoroServo.write(SERVO_1, 2000); /* set SERVO_1 on position 2000 (2mS pulse)*/

   EmoroServo.write(SERVO_0, EmoroServo.read(SERVO_1));   /*  SERVO_0 set to the position of SERVO_1  */

   Serial.print("read(SERVO_0) = ");           /* send position of SERVO_0 output            */
   Serial.println(EmoroServo.read(SERVO_0));
   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(0, 0);                        /* set LCD print location to 0, 0                */
      Lcd.print("EmoroServo.read");            /* print Examples function                     */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0                */
      Lcd.println(EmoroServo.read(SERVO_0));   /* print position of SERVO_0 motor                  */
   }
}

void loop(void){                            /* Arduino loop                                        */
}
```

```
float ReadPowerSupply(void);
```

Function reads input power supply of EMoRo 2560 controller.

Function prototype:
    arduino.h

Arguments:
    None

Return value:
    float – Result of function:
        (0 - 15V) – Power supply.

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                                /* Arduino setup                                    */
   Serial.begin(9600);                                 /* initialize UART 9600 bps                    */
   Serial.println("Example: ReadPowerSupply();");   /* send Examples name                          */
}

void loop(void){                                 /* Arduino loop                                     */

   Serial.print("Power Supply = ");              /* send Examples name                               */
   Serial.println(ReadPowerSupply());            /* send power supply of EMoRo 2560 controller          */

   if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(0, 0);                          /* set LCD print location to 0, 0                      */
      Lcd.print("U = ");                         /* print power supply of EMoRo 2560 controller         */
      Lcd.print(ReadPowerSupply());
      Lcd.print("  ");
   }
   delay(500);                                   /* wait 500 ms                                      */
}
```

```
int Ultrasonic.attach(unsigned char port);
```

Function initialize ultrasonic sensor on ports GPP_0 – GPP_7. Each initialized ultrasonic sensor needs 60 ms to measure distance, and if application program initialize all eight ultrasonic sensors refresh rate for each sensor will be 8*60 ms = 480 ms. To increase refresh rate of one sensor it is necessary to release resources of all unused sensors.

Function prototype:
```
      emoro_ultrasonic.h
```

Arguments:
```
      unsigned char port –  Ultrasonic [GPP_0 – GPP_7]
```

Return value:
```
      int – Result of function:
            (0)   – Ultrasonic sensor successfully initialized
            (-1)  – Error: Argument „port“ is out of range: [GPP_0 – GPP_7]
```

Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                              /* Arduino setup                                    */

   Serial.begin(9600);                               /* initialize UART 9600 bps                 */
   Serial.println("Example: Ultrasonic.attach();"); /* send Examples name                        */

   Ultrasonic.attach(GPP_0);                   /* initialize ultrasonic sensor on port GPP_0     */
}

void loop(void){                               /* Arduino loop                                     */
   int ultrasonic;

   ultrasonic = Ultrasonic.read(GPP_0);        /* read the value of ultrasonic sensor GPP_0       */

   Serial.print("Ultrasonic GPP_0 = ");        /* send name of ultrasonic sensor GPP_0             */
   Serial.println(ultrasonic);                 /* send value of ultrasonic sensor GPP_0      */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(0, 0);                        /* set LCD print location to 0, 0                  */
      Lcd.print("Ultrasonic GPP_0");           /* print initialized ultrasonic sensor     */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0                */
      Lcd.print(ultrasonic);                   /* print the value of ultrasonic sensor GPP_0      */
      Lcd.print("    ");                       /* delete LCD value of the previous print           */
   }
   delay(100);                                 /* wait 100 ms                                      */
}
```

```
int Ultrasonic.detach(unsigned char port);
```

Function releases resources of ultrasonic sensor on ports GPP_0 – GPP_7.

Function prototype:
```
      emoro_ultrasonic.h
```

Arguments:
```
      unsigned char port –  Ultrasonic [GPP_0 – GPP_7]
```

Return value:
```
      int – Result of function:
            (0)   – Resources of ultrasonic sensor successfully released
```

```
        (-1)  – Error: Argument „port“ is out of range: [GPP_0 – GPP_7]
```

Supported moduls:

```
    EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                  /* Arduino setup                                   */

   Serial.begin(9600);                                  /* initialize UART 9600 bps               */
   Serial.println("Example: Ultrasonic.detach();"); /* send Examples name                     */

   Ultrasonic.attach(GPP_0);                       /* initialize ultrasonic sensor on port GPP_0    */
   Ultrasonic.attach(GPP_1);                       /* initialize ultrasonic sensor on port GPP_1    */

   /* release resources of ultrasonic sensor GPP_1 to increase reading speed of sensor GPP_0    */
   Ultrasonic.detach(GPP_1);
}

void loop(void){                                   /* Arduino loop                                    */
   int ultrasonic;

   ultrasonic = Ultrasonic.read(GPP_0);            /* read ultrasonic sensor on port GPP_0              */

   Serial.print("Ultrasonic GPP_0 = ");            /* send name of ultrasonic sensor GPP_0              */
   Serial.println(ultrasonic);                     /* send value of ultrasonic sensor GPP_0      */

   if(ReadEmoroHardware() & LCD_AVAILABLE){        /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(0, 0);                            /* set LCD print location to 0, 0                    */
      Lcd.print("Ultrasonic GPP_0");               /* print initialized ultrasonic sensor GPP_0        */
      Lcd.locate(1, 0);                            /* set LCD print location to 1, 0               */
      Lcd.print(ultrasonic);                       /* print value of ultrasonic sensor GPP_0       */
      Lcd.print("    ");                           /* delete LCD value of the previous print          */
   }
   delay(100);                                     /* wait 100 ms                                       */
}
```

```
int Ultrasonic.read(unsigned char port);
```

Function reads ultrasonic sensor on ports GPP_0 - GPP_7. Before using this function ultrasonic sensor must be initialized with **Ultrasonic.attach();** function.

Function prototype:

```
    emoro_ultrasonic.h
```

Arguments:

```
    unsigned char port –  Ultrasonic [GPP_0 – GPP_7]
```

Return value:

```
    int – Result of function:
        (0-399) –  Sensor distance in cm
        (400)   –  Sensor out of range
        (-1)    –  Error: Argument „port“ out of range: [GPP_0 – GPP_7]
        (-2)    –  Error: Ultrasonic sensor is not initialized
```

Supported moduls:

```
    EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                  /* Arduino setup                                   */
```

```
   Serial.begin(9600);                                /* initialize UART 9600 bps                */
   Serial.println("Example: Ultrasonic.read();");     /* send Examples name                      */

   Ultrasonic.attach(GPP_0);                     /* initialize ultrasonic sensor on port GPP_0    */
}

void loop(void){                                 /* Arduino loop                                       */
   int ultrasonic;

   ultrasonic = Ultrasonic.read(GPP_0);          /* read ultrasonic sensor on port GPP_0               */

   Serial.print("Ultrasonic GPP_0 = ");          /* send name of ultrasonic sensor GPP_0               */
   Serial.println(ultrasonic);                   /* send value of ultrasonic sensor GPP_0      */

   if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(0, 0);                          /* set LCD print location to 0, 0                     */
      Lcd.print("Ultrasonic GPP_0");             /* print initialized ultrasonic sensor GPP_0     */
      Lcd.locate(1, 0);                          /* set LCD print location to 1, 0                */
      Lcd.print(ultrasonic);                     /* print value of ultrasonic sensor GPP_0        */
      Lcd.print("    ");                         /* delete LCD value of the previous print             */
   }
   delay(100);                                   /* wait 100 ms                                        */
}
```

```
int Lcd.init(void);
```

Function initialize LCD 2x16 of EMoRo GLAM module. Before **setup();** and **loop();** Arduino IDE will initialize LCD so re-initialization is not required.

Function prototype:
    emoro_lcd.h

Arguments:
    None

Return value:
```
int – Result of function:
     (0)    –  Lcd successfully initialized
     (-1)   –  Error: Unsuccessful communication with LCD
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                           /* Arduino setup                                */

  /* LCD is initialized so re-initialization is not required                          */

  Serial.begin(9600);                            /* initialize UART 9600 bps                */
  Serial.println("Example: Lcd.init();");        /* send Examples name                      */

  if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("Lcd.init();");                 /* print Examples name                          */
    Serial.println("Lcd Available");          /* send „Lcd Available“                */
  }
  else
    Serial.println("Lcd Not Available");      /* send „Lcd Not Available“              */
}

void loop(void){                            /* Arduino loop                                 */
}
```

```
int Lcd.clear(void);
```

Function clears LCD 2x16 of EMoRo GLAM module. After clearing the LCD position for printing is set to 0, 0 (first row, first column).

Function prototype:
    emoro_lcd.h

Arguments:
    None

Return value:
```
int – Result of function:
     (0)    –  LCD successfully cleared
     (-1)   –  Error: Unsuccessful communication with LCD
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                           /* Arduino setup                                */

  Serial.begin(9600);                       /* initialize UART 9600 bps                */
```

```
   Serial.println("Example: Lcd.clear();");    /* send Examples name                         */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");         /* send „Lcd Available“                  */
      Lcd.print("Lcd.clear();");               /* print Examples name                        */
      delay(2000);                             /* wait 2000 ms                           */
      Lcd.clear();                             /* clear LCD                                  */
   }
   else
      Serial.println("Lcd Not Available");     /* send „Lcd Not Available“               */
}

void loop(void){                               /* Arduino loop                                */
}
```

```
int Lcd.printString(char *str);
```

Function prints a string of characters on LCD 2x16 of EMoRo GLAM module. Max string length is 16 data bytes + 1 end of string byte.

Function prototype:
```
      emoro_lcd.h
```

Arguments:
```
      char *str     - Pointer to the first character of string
```

Return value:
```
      int - Result of function:
            (0)     -  String of characters successfully printed
            (-1)    -  Error: Unsuccessful communication with LCD
```
Supported moduls:
```
      EMoRo 2560, Extend board - EMoRo GLAM
```

Example:
```
void setup(void){                              /* Arduino setup                               */

   Serial.begin(9600);                                /* initialize UART 9600 bps            */
   Serial.println("Example: Lcd.printString();");    /* send Examples name                  */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");         /* send „Lcd Available“                  */
      Lcd.printString("printString();");       /* print Examples name                        */
   }
   else
      Serial.println("Lcd Not Available");     /* send „Lcd Not Available“               */
}

void loop(void){                               /* Arduino loop                                */
}
```

```
int Lcd.printChar(char data);
```

Function prints a single character on LCD 2x16 of EMoRo GLAM module.

Function prototype:
```
      emoro_lcd.h
```

Arguments:
```
      char data     - Character
```

Return value:

```
     int – Result of function:
          (0)     –  Character successfully printed
          (-1)    –  Error: Unsuccessful communication with LCD
```

Supported moduls:

```
     EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                              /* Arduino setup                                    */

   Serial.begin(9600);                               /* initialize UART 9600 bps                 */
   Serial.println("Example: Lcd.printChar();");      /* send Examples name                       */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");         /* send „Lcd Available“                    */
      Lcd.printChar('A');                      /* print character A                                  */
      Lcd.printChar('B');                      /* print character B                                  */
      Lcd.printChar('C');                      /* print character C                                  */
   }
   else
      Serial.println("Lcd Not Available");     /* send „Lcd Not Available“                   */
}

void loop(void){                               /* Arduino loop                                     */
}
```

```
long Lcd.print(val);
long Lcd.print(val, format);
```

Function prints ASCII text on LCD 2x16 of EMoRo GLAM module and can take many forms. Numbers are printed as ASCII character for each digit, decimal number are printed with 2 decimal places, a byte is printed as a single ASCII character and strings are printed until the \0 symbol.

Example:

Lcd.print(78);	- prints “78“
Lcd.print(1.23456) ;	- prints “1.23“
Lcd.print('N') ;	- prints 'N'
Lcd.print(“Hello World“) ;	- prints “Hello World“

As an option for printing can be used some other parameter that defines format of printed numbers (BIN, OCT, DEC, HEX).

Example:

Lcd.print(78, BIN);	- prints “1001110“
Lcd.print(78, OCT);	- prints “116“
Lcd.print(78, DEC);	- prints “78“
Lcd.print(78, HEX);	- prints “4E“
Lcd.print(1.23456, 0);	- prints “1“
Lcd.print(1.23456, 2);	- prints “1.23“
Lcd.print(1.23456, 4);	- prints “1.2346“

Function prototype:

```
     emoro_lcd.h
```

Arguments:

```
     val     – print data (any data type)
     format  – format of printed numbers (BIN, OCT, DEC, HEX)
```

Return value:

```
      long     – Function returns number of printed charactes
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                               /* Arduino setup                                        */

   Serial.begin(9600);                          /* initialize UART 9600 bps                       */
   Serial.println("Example: Lcd.print();");     /* send Examples name                             */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");          /* send „Lcd Available“                      */
      Lcd.print('P');                           /* print character P                                  */
      Lcd.print("I = ");                        /* print string “I = “                                */
      Lcd.print(3.14);                          /* print 3.14                                    */
   }
   else
      Serial.println("Lcd Not Available");      /* send „Lcd Not Available“                     */
}

void loop(void){                                /* Arduino loop                                         */
}
```

```
long Lcd.write(val);
long Lcd.write(str);
long Lcd.write(buf, len);
```

Function writes a byte or set of bytes on LCD 2x16 of EMoRo GLAM module. For printing ASCII characters and numbers, use the **Lcd.print ();** function.

Function prototype:

```
      emoro_lcd.h
```

Arguments:

```
      val      –  value to send as one byte
      str      –  string to send as a series of bytes
      buf      –  field for sending a series of bytes
      len      –  length of the field
```

Return value:

```
      long     – Function returns number of printed charactes
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                               /* Arduino setup                                        */

   Serial.begin(9600);                          /* initialize UART 9600 bps                       */
   Serial.println("Example: Lcd.write();");     /* send Examples name                             */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");          /* send „Lcd Available“                      */
      Lcd.write(72);                            /* print byte 72 (ASCII 72 = H)                         */
      Lcd.write("ELLO");                        /* print string „ELLO“                                */
   }
   else
      Serial.println("Lcd Not Available");      /* send „Lcd Not Available“                     */
}
```

```
void loop(void){                             /* Arduino loop                                         */
}
```

```
int Lcd.locate(unsigned char r, unsigned char c);
```

Function sets location for printing on LCD 2x16 of EMoRo GLAM module.

Function prototype:

```
     emoro_lcd.h
```

Arguments:

```
     unsigned char r      - Row
     unsigned char c      - Column
```

Return value:

```
     int - Result of function:
           (0)       -  Location for printing on LCD is successfully located
           (-1)      -  Error: Unsuccessful communication with LCD
```

Supported moduls:

```
     EMoRo 2560, Extend board - EMoRo GLAM
```

Example:

```
void setup(void){                            /* Arduino setup                                        */

   Serial.begin(9600);                       /* initialize UART 9600 bps                        */
   Serial.println("Example: Lcd.locate();"); /* send Examples name                              */

   if(ReadEmoroHardware() & LCD_AVAILABLE){  /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");       /* send „Lcd Available“                       */
      Lcd.locate(0, 0);                      /* set LCD print location to 0, 0                */
      Lcd.print("Hello");                    /* print „Hello“ on 0, 0                      */
      Lcd.locate(1, 0);                      /* set LCD print location to 1, 0                */
      Lcd.print("World");                    /* print „World“ on 1, 0                      */
   }
   else
      Serial.println("Lcd Not Available");   /* send „Lcd Not Available“                      */
}

void loop(void){                             /* Arduino loop                                         */
}
```

```
int Lcd.contrast(unsigned char con);
```

Function sets the contrast of LCD 2x16.

Function prototype:

```
     emoro_lcd.h
```

Arguments:

```
     unsigned char con    - Contrast
```

Return value:

```
     int - Result of function:
           (0)       -  Location for printing on LCD is successfully located
           (-1)      -  Error: Unsuccessful communication with LCD
```

Supported moduls:

EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                          /* Arduino setup                                  */

   Serial.begin(9600);                      /* initialize UART 9600 bps                    */
   Serial.println("Example: Lcd.locate();"); /* send Examples name                         */

   if(ReadEmoroHardware() & LCD_AVAILABLE){  /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");        /* send „Lcd Available“                     */
      Lcd.contrast(40);                        /* set contrast to 40                        */
      Lcd.locate(0, 0);                        /* set LCD print location to 0, 0            */
      Lcd.print("Hello");                      /* print „Hello“ on 0, 0                    */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0            */
      Lcd.print("World");                      /* print „World“ on 1, 0                    */
   }
   else
      Serial.println("Lcd Not Available");     /* send „Lcd Not Available“                  */
}

void loop(void){                           /* Arduino loop                                   */
}
```

`void Lcd.backlightOn(void);`

Function turns ON backlight on the LCD. After initialization of EMoRo controller default state of backlight is "ON".

Function prototype:
emoro_lcd.h

Arguments:
None

Return value:
None

Supported moduls:
EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                          /* Arduino setup                                  */

   /* After initialization of EMoRo controller default state of backlight is "ON"    */

   Serial.begin(9600);                          /* initialize UART 9600 bps                */
   Serial.println("Example: Lcd.backlightOn();"); /* send Examples name                    */

   if(ReadEmoroHardware() & LCD_AVAILABLE){  /* if LCD on EMoRo GLAM is initialized   */
      Serial.println("Lcd Available");        /* send „Lcd Available“                     */
      Lcd.backlightOff();                      /* turn backlight OFF                        */
      Lcd.print("Backlight Off ");             /* print "Backlight Off "          */
      delay(2000);                             /* wait 2000ms                                    */
      Lcd.locate(0, 0);                        /* set LCD print location to 0, 0        */
      Lcd.print("Backlight On ");              /* print "Backlight On "         */
      Lcd.backlightOn();                       /* turn backlight ON                        */
   }
   else
```

```
    Serial.println("Lcd Not Available");     /* send „Lcd Not Available“                   */
}

void loop(void){                             /* Arduino loop                                     */
}
```

```
void Lcd.backlightOff(void);
```

Function turns OFF backlight on the LCD. After initialization of EMoRo controller default state of backlight is "ON".

Function prototype:
    emoro_lcd.h

Arguments:
    None

Return value:
    None

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                            /* Arduino setup                                    */

  /* After initialization of EMoRo controller default state of backlight is "ON"      */

  Serial.begin(9600);                             /* initialize UART 9600 bps                  */
  Serial.println("Example: Lcd.backlightOff();"); /* send Examples name                        */

  if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
    Serial.println("Lcd Available");          /* send „Lcd Available“                      */
    Lcd.backlightOff();                       /* turn backlight OFF                             */
    Lcd.print("Backlight Off ");              /* print "Backlight Off "           */
  }
  else
    Serial.println("Lcd Not Available");     /* send „Lcd Not Available“                   */
}

void loop(void){                             /* Arduino loop                                     */
}
```

```
int Acc.init(void);
```

Function initialize Accelerometer on EMoRo GLAM module. Before **setup();** and **loop();** Arduino IDE will initialize Accelerometer so re-initialization is not required.

Function prototype:

    emoro_acc.h

Arguments:

    None

Return value:

```
int - Result of function:
      (0)     -  Accelerometer successfully initialized
      (-1)    -  Error: Unsuccessful communication with accelerometer
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                 /* Arduino setup                                        */

   /* Accelerometer is initialized so re-initialization is not required     */

   Serial.begin(9600);                            /* initialize UART 9600 bps                       */
   Serial.println("Example: Acc.init();");        /* send Examples name                             */
   if(ReadEmoroHardware() & ACC_AVAILABLE)        /* if Accelerometer is initialized                   */
      Serial.println("Acc Available");            /* send „Acc Available“                     */
   else
      Serial.println("Acc Not Available");        /* send „Acc Not Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){       /* if LCD is initialized   */
      Lcd.print("Acc.init();");                   /* print Examples name                               */
      Lcd.locate(1, 0);                           /* set LCD print location to 1, 0             */
      if(ReadEmoroHardware() & ACC_AVAILABLE)     /* if Accelerometer is initialized                      */
         Lcd.print("Available");                  /* print „Available“                  */
      else
         Lcd.print("Not Available");              /* print „Not Available“                 */
   }
}

void loop(void){                                  /* Arduino loop                                         */
}
```

```
unsigned char Acc.testConnection(void);
```

Function test availability of accelerometer on EMoRo GLAM module.

Function prototype:

    emoro_acc.h

Arguments:

    None

Return value:

```
unsigned char    - Result of function:
                 (0)   -  Accelerometer not available
                 (1)   -  Accelerometer available
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                   /* Arduino setup                                   */

   Serial.begin(9600);                                       /* initialize UART 9600 bps              */
   Serial.println("Example: Acc.testConnection();");   /* send Examples name                    */
   if(Acc.testConnection())                        /* if accelerometer is available                   */
      Serial.println("Acc Available");             /* send „Acc Available“                    */
   else
      Serial.println("Acc Not Available");         /* send „Acc Not Available“           */

   if(ReadEmoroHardware() & LCD_AVAILABLE){        /* if LCD is initialized   */
      Lcd.print("Acc.connection()");               /* print Examples name                          */
      Lcd.locate(1, 0);                            /* set LCD print location to 1, 0              */
      if(Acc.testConnection())                     /* if accelerometer is available                */
         Lcd.print("Available");                   /* print „Available“                */
      else
         Lcd.print("Not Available");               /* print „Not Available“              */
   }
}

void loop(void){                                   /* Arduino loop                                    */
}
```

```
int Acc.read(int *x, int *y, int *z);
```

Function reads 3-axis of accelerometer (x, y, z) and returns the 10-bit value written as II complement -512 to +511 (-2.000g – 1.996g) (g=9.81m/s^2).

Function prototype:

```
      emoro_acc.h
```

Arguments:

```
      int *x   – a pointer to the „x“ axis acceleration
      int *y   – a pointer to the „y“ axis acceleration
      int *z   – a pointer to the „z“ axis acceleration
```

Return value:

```
      int   – Result of function:
               (0)      –  Axis of accelerometer successfully readed
               (-1)     –  Error: Unsuccessful communication with accelerometer
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                   /* Arduino setup                                   */

   Serial.begin(9600);                             /* initialize UART 9600 bps                      */
   Serial.println("Example: Acc.read();");         /* send Examples name                            */
   if(Acc.testConnection() == 0)                   /* if accelerometer not available                  */
      Serial.println("Acc Not Available");         /* send „Acc Not Available“                  */

   if(ReadEmoroHardware() & LCD_AVAILABLE){        /* if LCD is initialized   */
      Lcd.print("Acc.read();");                    /* print Examples name                          */
      if(Acc.testConnection() == 0){               /* if accelerometer not available                  */
         Lcd.locate(1, 0);                         /* set LCD print location to 1, 0              */
         Lcd.print("Not Available");               /* print „Accelerometer Not Available“          */
      }
   }
}

void loop(void){                                 /* Arduino loop                                       */
   int x, y, z;                                  /* variables for saving acceleration as II complement    */
   int res;                                      /* variable for saving result of function
                                                                                                            */
   float x_phy, y_phy, z_phy;                    /* variables for axis of acceleration          */

   res = Acc.read(&x, &y, &z);                   /* read x, y and z axis of acceleration                */

   if(res == 0){                                 /* if axis of accelerometer are successfully readed    */
      char buf[32];                              /* send axis of acceleration to UART0                 */
      sprintf(buf, "X =%4d, Y =%4d, Z =%4d", x, y, z);
      Serial.println(buf);

      x_phy=x*2.0*9.81/512;                      /* conversion of acceleration from II complement    */
      y_phy=y*2.0*9.81/512;
      z_phy=z*2.0*9.81/512;

      Serial.print("X =");
      Serial.print(x_phy);
      Serial.print(" m/s^2, Y =");
      Serial.print(y_phy);
      Serial.print(" m/s^2, Z =");
      Serial.println(z_phy);

      if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
         Lcd.locate(1, 0);                       /* set LCD print location to 1, 0            */
         sprintf(buf,"X =%4d Y =%4d", x, y);    /* print x and y axis of accelerometer on LCD         */
         Lcd.print(buf);
      }
   }
   delay(300);                                   /* wait 300 ms                                        */
}
```

```
int Acc.readX(void);
int Acc.readY(void);
int Acc.readZ(void);
```

Function reads acceleration of the X, Y or Z axis of accelerometer on EMoRo GLAM module and returns the 10-bit value written as II complement -512 to +511 (-2.000g – 1.996g) (g=9.81m/s^2).

Function prototype:
      emoro_acc.h

Arguments:
      None

Return value:
      int - Result of function: acceleration of axis that is reading (as II complement)

Supported moduls:
      EMoRo 2560, Extend board - EMoRo GLAM

Example za X os:

```
void setup(void){                                /* Arduino setup                                      */

   Serial.begin(9600);                           /* initialize UART 9600 bps                    */
   Serial.println("Example: Acc.readX();");    /* send Examples name                          */
   if(Acc.testConnection() == 0)               /* if accelerometer not available                    */
      Serial.println("Acc Not Available");     /* send „Acc not available“                   */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD is initialized   */
      Lcd.print("Acc.readX();");               /* print Examples name                               */
      if(Acc.testConnection() == 0){           /* if accelerometer not available                    */
         Lcd.locate(1, 0);                     /* set LCD print location to 1, 0             */
         Lcd.print("Not Available");           /* print „Not Available“               */
      }
   }
}

void loop(void){                               /* Arduino loop                                      */

   if(Acc.testConnection()){                   /* if accelerometer is available                     */
      char buf[32];
      int x = Acc.readX();                     /* read X axis of acceleration                          */

      sprintf(buf, "X =%4d", x);
      Serial.println(buf);                     /* send X axis to UART0                   */

      if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
         Lcd.locate(1, 0);                     /* set LCD print location to 1, 0             */
         Lcd.print(buf);                       /* print X axis on LCD                    */
      }
   }
   delay(300);                                 /* wait 300 ms                                       */
}
```

```
int Gyr.initBasic(void);
```

Function initialize Gyroscope on EMoRo GLAM module in Basic Mode. Before **setup();** and **loop();** Arduino IDE will initialize Gyroscope so re-initialization is not required. In Basic mode is allowed reading the angular acceleration of all three axes.

Function prototype:
```
      emoro_gyr.h
```

Arguments:
```
      None
```

Return value:
```
      int – Result of function:
               (0)   –  Gyroscope successfully initialized in Basic mode
               (-1)  –  Error: Unsuccessful communication with gyroscope
```
Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                               /* Arduino setup                                        */

   /* Gyroscope is initialized so re-initialization is not required                               */

   Serial.begin(9600);                          /* initialize UART 9600 bps                            */
   Serial.println("Example: Gyr.init();");      /* send Examples name                                  */
   if(ReadEmoroHardware() & GYR_AVAILABLE)      /* if gyroscope is initialized                            */
      Serial.println("Gyr Available");          /* send „GYR Available“                          */
   else
      Serial.println("Gyr Not Available");      /* send „GYR Not Available“                         */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Gyr.init();");                 /* print Examples name                                    */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0               */
      if(ReadEmoroHardware() & GYR_AVAILABLE)   /* if gyroscope is available                                */
         Lcd.print("Available");                /* print „Gyroscope Available“                              */
      else
         Lcd.print("Not Available");            /* print „Gyroscope Not Available“                          */
   }
}

void loop(void){                                /* Arduino loop                                             */
}
```

```
int Gyr.initBasic(unsigned int odr, unsigned int range);
```

Function initialize Gyroscope on EMoRo GLAM module in Basic Mode with specified parameters. In Basic mode is allowed reading the angular acceleration of all three axes. When calling the function is necessary to specify the "output data rate" and scale of reading gyroscope as "range".

Function prototype:
```
      emoro_gyr.h
```

Arguments:
```
      unsigned int odr    – output data rate possible values:
                            95, 190, 380 i 760 [Hz] (readings per second)
      unsigned int range – possible values of scale of reading: 250, 500 i 2000 (degrees per second)
```

Return value:

```
    int – Result of function:
          (0)  –  Gyroscope successfully initialized in Basic mode with specified parameters
          (-1) –  Error: Wrong Parameters
          (-2) –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                               /* Arduino setup                                          */

  /* Gyroscope is initialized so re-initialization is not required                              */

  Serial.begin(9600);                           /* initialize UART 9600 bps                          */
  Serial.println("Example: Gyr.init();");       /* send Examples name                                */
  if(ReadEmoroHardware() & GYR_AVAILABLE)       /* if gyroscope is initialized                          */
    Serial.println("Gyr Available");            /* send „GYR Available“                         */
  else
    Serial.println("Gyr Not Available");        /* send „GYR Not Available“                        */

  if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
    Gyr.initBasic(190, 500);                    /* initialization of gyroscope in Basic mode with settings:
                                                   generating data 190 times per second and the range of
                                                   measurement of +- 500°/second       */

    Lcd.print("Gyr.init();");                   /* print Examples name                                  */
    Lcd.locate(1, 0);                           /* set LCD print location to 1, 0                 */
    if(ReadEmoroHardware() & GYR_AVAILABLE)     /* if gyroscope is available                              */
      Lcd.print("Available");                   /* print „Gyroscope je Available“                           */
    else
      Lcd.print("Not Available");               /* print „Gyroscope Not Available“                        */
  }
}

void loop(void){                                /* Arduino loop                                           */
}
```

```
int Gyr.init(void);
```

Function initialize Gyroscope on EMoRo GLAM module in Advanced mode. In Advance mode is allowed reading the turning angle for all three axes. The integration of angular velocity by time is done automatically in the interrupt routine.

Function prototype:

    emoro_gyr.h

Arguments:

    None

Return value:

```
    int – Result of function:
          (0)  –  Gyroscope successfully initialized in Advanced mode
          (-1) –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                               /* Arduino setup                                          */

  /* Gyroscope Gyroscope is initialized so re-initialization is not required                      */
```

```
   Serial.begin(9600);                              /* initialize UART 9600 bps                     */
   Serial.println("Example: Gyr.init();");          /* send Examples name                           */
   if(ReadEmoroHardware() & GYR_AVAILABLE)          /* if gyroscope is initialized                    */
      Serial.println("Gyr Available");              /* send „GYR Available“                   */
   else
      Serial.println("Gyr Not Available");          /* send „GYR nije Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){         /* if LCD on EMoRo GLAM is initialized   */
      Gyr.init();                                   /* initialization of gyroscope in Advanced mode          */
      Lcd.print("Gyr.init();");                     /* print Examples name                            */
      Lcd.locate(1, 0);                             /* set LCD print location to 1, 0            */
      if(ReadEmoroHardware() & GYR_AVAILABLE)       /* if gyroscope is available                        */
         Lcd.print("Available");                    /* print „Gyroscope Available“                      */
      else
         Lcd.print("Not Available");                /* print „Gyroscope Not Available“                  */
   }
}

void loop(void){                                    /* Arduino loop                                     */
}
```

```
int Gyr.init(unsigned int odr, unsigned int range);
```

Function initialize Gyroscope on EMoRo GLAM module in Advanced mode with specified parameters. In Advanced mode is allowed reading the turning angle for all three axes. When calling the function is necessary to specify the "output data rate" and scale of reading gyroscope as "range". The integration of angular velocity by time is done automatically in the interrupt routine.

Function prototype:

```
      emoro_gyr.h
```

Arguments:

```
      unsigned int odr   - output data rate possible values:
                            95, 190, 380 i 760 [Hz] (readings per second)
      unsigned int range - possible values of scale of reading: 250, 500 i 2000 (degrees per second)
```

Return value:

```
      int - Result of function:
               (0)   -  Gyroscope successfully initialized in Advanced mode with specified parameters
               (-1)  -  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

```
      EMoRo 2560, Extend board - EMoRo GLAM
```

Example:

```
void setup(void){                                   /* Arduino setup                                    */

   /* Gyroscope is initialized so re-initialization is not required                            */

   Serial.begin(9600);                              /* initialize UART 9600 bps                     */
   Serial.println("Example: Gyr.init();");          /* send Examples name                           */
   if(ReadEmoroHardware() & GYR_AVAILABLE)          /* if gyroscope is initialized                    */
      Serial.println("Gyr Available");              /* send „GYR Available“                   */
   else
      Serial.println("Gyr Not Available");          /* send „GYR nije Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){         /* if LCD on EMoRo GLAM is initialized   */
      Gyr.init(190, 500);                        /* initialization of gyroscope in Advanced mode with settings:
                                                         generating data 190 times per second and the range of
```

```
                                                   measurement of +- 500°/second      */
      Lcd.print("Gyr.init();");                 /* print Examples name                         */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0         */
      if(ReadEmoroHardware() & GYR_AVAILABLE)   /* if gyroscope is available                         */
         Lcd.print("Available");                /* print „Gyroscope Available“                       */
      else
         Lcd.print("Not Available");            /* print „Gyroscope Not Available“                   */
   }
}

void loop(void){                                /* Arduino loop                                      */
}
```

```
int Gyr.stop(void);
```

Function stops the Advanced mode of gyroscope. Because the numerical integration and filtering (used for calculating the angular position of gyroscope) are quite demanding for the microcontroller, it is recommended to stop the Advanced mode of gyroscope when is unnecessary, and thus save controllers time for data and interrupt processing.

Function prototype:

      emoro_gyr.h

Arguments:

      None

Return value:

```
      int – Result of function:
            (0)   –  Advanced mode of gyroscope successfully stopped
            (-1)  –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

      EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                               /* Arduino setup                                     */

   /* Gyroscope is initialized so re-initialization is not required                        */

   Serial.begin(9600);                          /* initialize UART 9600 bps                    */
   Serial.println("Example: Gyr.init();");      /* send Examples name                          */
   if(ReadEmoroHardware() & GYR_AVAILABLE)      /* if gyroscope is initialized                       */
      Serial.println("Gyr Available");          /* send „GYR Available“                  */
   else
      Serial.println("Gyr Not Available");      /* send „GYR Not Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Gyr.init(190, 500);                     /* initialization of gyroscope in Advanced mode with settings:
                                                   generating data 190 times per second and the range of
                                                   measurement of +- 500°/second      */
      Lcd.print("Gyr.init();");                 /* print Examples name                         */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0         */
      if(ReadEmoroHardware() & GYR_AVAILABLE){  /* if gyroscope is available                         */
         Lcd.print("Available");                /* print „Gyroscope Available“                       */
      }
      else
         Lcd.print("Not Available");            /* print „Gyroscope Not Available“                   */
   }
}

void loop(void){                                /* Arduino loop                                      */
```

```
   int res;                                 /* variable for return value of function           */
   double x_deg, y_deg, z_deg;              /* angular position for each axis                    */
   char buf[64];                            /* additional buffer for printing                    */

   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position for all three axes        */

   if(res == 0){                            /* if it is successfully readed print message    */
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
   }
  else
     Serial.println("Can't read angular position."); /* message for unseccessfully reading of angular
position*/

  if( (ReadEmoroHardware() & LCD_AVAILABLE) && (res==0) ){
       Lcd.locate(0, 0);                    /* set LCD print location to 0, 0        */
       Lcd.print(" X    Y    Z");           /* print names of axes                                 */
       Lcd.locate(1, 0);                    /* set LCD print location to 1, 0     */
       sprintf(buf, "%3d %3d %3d", (int)x_deg, (int)y_deg, (int)z_deg);  /* print angular position       */
       Lcd.print(buf);
  }
  else if(ReadEmoroHardware() & LCD_AVAILABLE){ /*message for unseccessfully reading of angular position*/
       Lcd.locate(1, 0);
       Lcd.print("Can't read      ");
  }

  if(ReadEmoroHardware() & SW_AVAILABLE){      /* if pushbuttons are available                       */
      if(ReadSwitch(SW_1)){                 /* by pressing the SW1 pushbutton current position of the */
      Serial.println("Reset current position. X=0, Y=0, Z=0."); /* gyroscope is set to zero on all axes */
         Gyr.resetDegrees();
     }
     else if(ReadSwitch(SW_2)){             /* by pressing the SW_2 pushbutton stops gyroscope in     */
         Gyr.stop();                        /* Advanced mode and thus save controllers time for    */
         Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing      */
     }
  }
  delay(300);                               /* wait 300 ms                            */
}
```

```
unsigned char Gyr.testConnection(void);
```

Function test availability of gyroscope on EMoRo GLAM module.

Function prototype:
    emoro_gyr.h

Arguments:
    None

Return value:
```
     unsigned char    –  Result of function:
                      (0)   –  Gyroscope not available
                      (1)   –  Gyroscope available
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                           /* Arduino setup                                     */
```

```
   Serial.begin(9600);                                  /* initialize UART 9600 bps              */
   Serial.println("Example: Gyr.testConnection();");   /* send Examples name                    */
   if(Gyr.testConnection())                       /* if gyroscope is available                         */
      Serial.println("Gyr Available");            /* send „GYR Available“                    */
   else
      Serial.println("Gyr Not Available");        /* send „GYR Not Available“                  */

   if(ReadEmoroHardware() & LCD_AVAILABLE){       /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Gyr.connection()");              /* print Examples name                           */
      Lcd.locate(1, 0);                           /* set LCD print location to 1, 0            */
      if(Gyr.testConnection())                    /* if gyroscope is available                       */
         Lcd.print("Available");                  /* print „Gyroscope Available“                     */
      else
         Lcd.print("Not Available");              /* print „Gyroscope Not Available“                 */
   }
}

void loop(void){                                  /* Arduino loop                                    */
}
```

```
int Gyr.readDegrees(double *x, double *y, double *z);
```

Function reads current angular position for all three axes. Before reading the gyroscope must be initialized in Advanced mode. Simultaneously moving of all three axes comes to drift of reading. These drift can be reduced by configuring the Advanced mode with different parameters, increasing the number of sampling during calibration, adjustment of filtration, ie, the configuration of gyroscope should be customized to the exact dynamic properties in which the gyroscope is used.

Function prototype:

```
      emoro_gyr.h
```

Arguments:

```
      double *x  – variable x will take the value of the angular position of the X axis (0-359⁰)
      double *y  – variable y will take the value of the angular position of the Y axis (0-359⁰)
      double *z  – variable z will take the value of the angular position of the Z axis (0-359⁰)
```

Return value:

```
      int – Result of function:
            (0)   –  Angular position successfully taken for all three axes
            (-1)  –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                 /* Arduino setup                                   */
   Serial.begin(9600);                            /* initialize UART 9600 bps                  */
   Serial.println("Example: Gyr.init();");        /* send Examples name                        */
   if(ReadEmoroHardware() & GYR_AVAILABLE)        /* if gyroscope is initialized                     */
      Serial.println("Gyr Available");            /* send „GYR Available“                    */
   else
      Serial.println("Gyr Not Available");        /* send „GYR Not Available“                  */

   if(ReadEmoroHardware() & LCD_AVAILABLE){       /* if LCD on EMoRo GLAM is initialized   */
      Gyr.init();                                 /* initialization of gyroscope in Advanced mode with */
                                                  /* Standard settings                               */
      Lcd.print("Gyr.init();");                   /* print Examples name                           */
      Lcd.locate(1, 0);                           /* set LCD print location to 1, 0            */
      if(ReadEmoroHardware() & GYR_AVAILABLE){ /* if gyroscope is available                       */
         Lcd.print("Available");                  /* print „Gyroscope Available“                     */
      }
      else
         Lcd.print("Not Available");            /* print „Gyroscope Not Available“                    */
   }
}

void loop(void){                                 /* Arduino loop                                       */
   int res;                                      /* return value of function            */
   double x_deg, y_deg, z_deg;                   /* angular position for each axis                       */
   char buf[64];                                 /* additional buffer for printing                       */

   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position for all three axes        */

   if(res == 0){                                 /* if it is successfully readed print massage    */
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
   }
  else
     Serial.println("Can't read angular position."); /* message for unseccessfully reading of angular  */
                                                      /* postion                                         */

  if( (ReadEmoroHardware() & LCD_AVAILABLE) && (res==0) ){
      Lcd.locate(0, 0);                          /* set LCD print location to 0, 0      */
      Lcd.print(" X    Y    Z");                 /* print names of axes                                  */
      Lcd.locate(1, 0);                          /* set LCD print location to 1, 0   */
      sprintf(buf, "%3d %3d %3d", (int)x_deg, (int)y_deg, (int)z_deg);  /* print angular position      */
      Lcd.print(buf);
  }
  else if(ReadEmoroHardware() & LCD_AVAILABLE){ /*message for unseccessfully reading of angular postion*/
      Lcd.locate(1, 0);
      Lcd.print("Can't read       ");
  }

  if(ReadEmoroHardware() & SW_AVAILABLE){       /* if pushbuttons are available                         */
      if(ReadSwitch(SW_1)){                      /* by pressing the SW1 pushbutton current position of the */
      Serial.println("Reset current position. X=0, Y=0, Z=0."); /* gyroscope is set to zero on all axes */
         Gyr.resetDegrees();
      }
     else if(ReadSwitch(SW_2)){                  /* by pressing the SW_2 pushbutton gyroscope in Advanced */
         Gyr.stop();                             /* mode stops and thus save microcontrollers time for    */
         Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing      */
    }
  }
  delay(300);                                    /* wait 300 ms                          */
}
```

```
double Gyr.readDegreesX(void);
double Gyr.readDegreesY(void);
double Gyr.readDegreesZ(void);
```

Function reads the current angular position of X, Y or Z axis. Before reading gyroscope must be initialized in Advanced mode.

Function prototype:

    emoro_gyr.h

Arguments:

    None

Return value:

    double – Result of function:

```
          (>=0) -  Angular position of axis that is readed
          (-1)  –  Error: Gyroscope not in Advanced mode
          (-2)  –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:

```
    EMoRo 2560, Extend board – EMoRo GLAM
```

Example za X os:

```
void setup(void){                              /* Arduino setup                                     */
  Serial.begin(9600);                          /* initialize UART 9600 bps                     */
  Serial.println("Example: Gyr.init();");      /* send Examples name                           */
  if(ReadEmoroHardware() & GYR_AVAILABLE)      /* if gyroscope is initialized                       */
    Serial.println("Gyr Available");           /* send „GYR Available“                    */
  else
    Serial.println("Gyr Not Available");       /* send „GYR Not Available“                    */

  if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
    Gyr.init();                                /* initialization of gyroscope in Advanced mode with */
                                               /* Standard settings                                */
    Lcd.print("Gyr.init();");                  /* print Examples name                              */
    Lcd.locate(1, 0);                          /* set LCD print location to 1, 0               */
    if(ReadEmoroHardware() & GYR_AVAILABLE){   /* if gyroscope is available                         */
      Lcd.print("Available");                  /* print „Gyroscope Available“                       */
    }
    else
      Lcd.print("Not Available");              /* print „Gyroscope Not Available“                   */
  }
}

void loop(void){                               /* Arduino loop                                      */
  double x_deg;                                /* angular position for X axis                       */
  char buf[64];                                /* addiotional buffer for printing                   */

  x_deg = Gyr.readDegreesX();                  /* reading the angular postion of X axis             */

  if(x_deg>=0){                                /* if it is successfully readed print message       */
    sprintf(buf, "Current position: X =%3d", (int)x_deg);
    Serial.println(buf);
  }
 else
   Serial.println("Can't read angular position."); /* message for unseccessfully reading of angular  */
                                                   /*position    */

 if( (ReadEmoroHardware() & LCD_AVAILABLE) && (x_deg>=0) ){
     Lcd.locate(0, 0);                         /* set LCD print location to 0, 0                    */
     Lcd.print(" X ");                         /* print X                                          */
     Lcd.locate(1, 0);                         /* set LCD print location to 1, 0                    */
     sprintf(buf, "%3d", (int)x_deg);          /* print angular position                            */
     Lcd.print(buf);
 }
 else if(ReadEmoroHardware() & LCD_AVAILABLE){/*message for unseccessfully reading of angular position*/
     Lcd.locate(1, 0);
     Lcd.print("Can't read      ");
 }

 if(ReadEmoroHardware() & SW_AVAILABLE){       /* if pushbuttons are available                       */
    if(ReadSwitch(SW_1)){                      /* by pressing the SW1 pushbutton gyroscope in Advanced  */
      Gyr.stop();                              /* mode stops and thus save microcontrollers time for    */
      Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing */
    }
 }
```

```
  delay(300);                                   /* wait 300 ms                           */
}
```

```
int Gyr.setDegrees(double x, double y, double z);
```

Function sets current angular position for all 3 axes to values that are in arguments of function.

Function prototype:
```
     emoro_gyr.h
```

Arguments:
```
     double x – setting X angular position on given value (0-359°)
     double y – setting Y angular position on given value (0-359°)
     double z – setting Z angular position on given value (0-359°)
```

Return value:
```
     int – Result of function:
             (0)   –  Angular position successfully seted
             (-1)  –  Error: Gyroscope not in Advanced mode
```

Supported moduls:
```
     EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                               /* Arduino setup                                   */
   Serial.begin(9600);                          /* initialize UART 9600 bps                   */
   Serial.println("Example: Gyr.init();");      /* send Examples name                         */
   if(ReadEmoroHardware() & GYR_AVAILABLE)      /* if gyroscope is initialized                     */
      Serial.println("Gyr Available");          /* send „GYR Available“                 */
   else
      Serial.println("Gyr Not Available");      /* send „GYR nije Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Gyr.init();                               /* initialization of gyroscope in Advanced mode with */
                                                /* Standard settings                             */
      Lcd.print("Gyr.init();");                 /* print Examples name                           */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0            */
      if(ReadEmoroHardware() & GYR_AVAILABLE){  /* if gyroscope is available                       */
         Lcd.print("Available");                /* print „Gyroscope Available“                     */
      }
      else
         Lcd.print("Not Available");            /* print „Gyroscope Not Available“                 */
   }
}

void loop(void){                                /* Arduino loop                                    */
   int res;                                     /* return value of function            */
   double x_deg, y_deg, z_deg;                  /* angular position for each axis                    */
   char buf[64];                                /* additional buffer for printing                    */

   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position for all three axes     */

   if(res == 0){                                /* if it is successfully readed print message    */
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
   }
  else
    Serial.println("Can't read angular position."); /* message for unsuccessfully reading of angular */
                                                    /*position                         */
  if( (ReadEmoroHardware() & LCD_AVAILABLE) && (res==0) ){
      Lcd.locate(0, 0);                         /* set LCD print location to 0, 0     */
      Lcd.print(" X    Y    Z");                /* print axes names                             */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0   */
      sprintf(buf, "%3d %3d %3d", (int)x_deg, (int)y_deg, (int)z_deg);  /* print angular position      */
      Lcd.print(buf);
  }
  else if(ReadEmoroHardware() & LCD_AVAILABLE){ /*message for unsuccessfully reading of angular position*/
      Lcd.locate(1, 0);
      Lcd.print("Can't read      ");
  }

  if(ReadEmoroHardware() & SW_AVAILABLE){       /* if pushbuttons are available                         */
     if(ReadSwitch(SW_1)){                      /* by pressing the SW1 current position of gyroscope  */
        Serial.println("Set current position. X=0, Y=180, Z=90.");  /* is set to: X=0, Y=180, Z=90   */
        Gyr.setDegrees(0, 180, 90);
     }
     else if(ReadSwitch(SW_2)){                 /* by pressing the SW2 pushbutton gyroscope in Advanced  */
        Gyr.stop();                             /* mode stops and thus save microcontrollers time for    */
        Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing */
    }
  }
  delay(300);                                   /* wait 300 ms                          */
}
```

```
int Gyr.resetDegrees(void);
```

Function sets current angular postition to 0 degress for all 3 axes.

Function prototype:

      emoro_gyr.h

Arguments:

      None

Return value:

```
      int – Result of function:
              (0)   –  Angular position successfully seted  for all 3 axes
              (-1)  –  Error: Gyroscope not in Advanced mode
```

Supported moduls:

      EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                               /* Arduino setup                                      */
   Serial.begin(9600);                          /* initialize UART 9600 bps                        */
   Serial.println("Example: Gyr.init();");      /* send Examples name                              */
   if(ReadEmoroHardware() & GYR_AVAILABLE)      /* if gyroscope is initialized                        */
      Serial.println("Gyr Available");          /* send „GYR Available“                        */
   else
      Serial.println("Gyr Not Available");      /* send „GYR Not Available“                      */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized    */
      Gyr.init();                               /* initialization of gyroscope in Advanced mode with */
                                                /* Standard settings                                 */
      Lcd.print("Gyr.init();");                 /* print Examples name                               */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0             */
      if(ReadEmoroHardware() & GYR_AVAILABLE){  /* if gyroscope is available                          */
         Lcd.print("Available");                /* print „Gyroscope Available“                        */
      }
      else
         Lcd.print("Not Available");           /* print „Gyroscope Not Available“                  */
   }
}

void loop(void){                               /* Arduino loop                                     */
   int res;                                    /* return value of function           */
   double x_deg, y_deg, z_deg;                 /* angular position for each axis                     */
   char buf[64];                               /* additional buffer for printing                     */

   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position for all 3 axes         */

   if(res == 0){
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
   }
  else
    Serial.println("Can't read angular position."); /*message for unseccessfully reading of angular pos*/

  if( (ReadEmoroHardware() & LCD_AVAILABLE) && (res==0) ){
      Lcd.locate(0, 0);                        /* set LCD print location to 0, 0     */
      Lcd.print(" X   Y   Z");                 /* print axes names                                 */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0   */
      sprintf(buf, "%3d %3d %3d", (int)x_deg, (int)y_deg, (int)z_deg);  /* print angular positions     */
      Lcd.print(buf);
  }
  else if(ReadEmoroHardware() & LCD_AVAILABLE){ /*message for unseccessfully reading of angular position*/
      Lcd.locate(1, 0);
      Lcd.print("Can't read      ");
  }

  if(ReadEmoroHardware() & SW_AVAILABLE){      /* if pushbuttons are available                       */
     if(ReadSwitch(SW_1)){                     /* by pressing the SW1 current position of gyroscope  */
        Serial.println("Reset current position. X=0, Y=0, Z=0.");  /* is setted to 0 on all 3 axes   */
        Gyr.resetDegrees();
     }
      else if(ReadSwitch(SW_2)){               /* by pressing the SW2 pushbutton gyroscope in Advanced  */
        Gyr.stop();                            /* mode stops and thus save microcontrollers time for    */
        Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing */
    }
  }
  delay(300);                                  /* wait 300 ms                            */
}
```

```
int Gyr.read(int *x, int *y, int *z);
```

Function reads 3 axes of gyroscope (x, y, z) and return 16 bit values written as II complement -32768 to +32767 (-250 – 250 degrees per second; dps).

Function prototype:

```
      emoro_gyr.h
```

Arguments:

```
      int *x  – pointer to value of „x“ axis angular acceleration
      int *y  – pointer to value of „y“ axis angular acceleration
      int *z  – pointer to value of „z“ axis angular acceleration
```

Return value:

```
      int   – Result of function:
        (0)     –  Angular acceleration of all 3 axes are successfully readed
        (-1)    –  Error: Unsuccessful communication with gyroscope
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                               /* Arduino setup                                      */

  Serial.begin(9600);                           /* initialize UART 9600 bps                      */
  Serial.println("Example: Gyr.read();");       /* send Examples name                            */
  if(Gyr.testConnection() == 0)                 /* if gyroscope not Available                         */
    Serial.println("Gyr Not Available");        /* send „GYR Not Available“                    */

  if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("Gyr.read();");                   /* print Examples name                                */
    if(Gyr.testConnection() == 0){              /* if gyroscope not Available                         */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0              */
      Lcd.print("Not Available");               /* print „Gyroscope Not Available“                    */
    }
  }
}

void loop(void){                                /* Arduino loop                                       */
  int x, y, z, res;

  res = Gyr.read(&x, &y, &z);                   /* read x, y and z axis of gyroscope                     */

  if(res == 0){                                 /* if axes are successfully readed             */
    char buf[64];                               /* send axes to UART0                    */
    sprintf(buf, "X =%6d, Y =%6d, Z =%6d", x, y, z);
    Serial.println(buf);

    if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0              */
      sprintf(buf,"X =%6d", x);                 /* print X axis of gyroscope on LCD                   */
      Lcd.print(buf);
    }
  }
  delay(300);                                   /* wait 300 ms                                        */
}
```

```
int Gyr.readX(void);
int Gyr.readY(void);
int Gyr.readZ(void);
```

Function reads angular acceleration of X, Y or Z axis of gyroscope and return 16 bit values written as II complement -32768 to +32767 (-250 – 250 dps).

Function prototype:
    emoro_gyr.h

Arguments:
    None

Return value:
    int – Result of function: Angular acceleration of gyroscope axis

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example za X os:

```
void setup(void){                               /* Arduino setup                                 */

   Serial.begin(9600);                          /* initialize UART 9600 bps                 */
   Serial.println("Example: Gyr.readX();");     /* send Examples name                       */
   if(Gyr.testConnection() == 0)                /* if gyroscope not Available                    */
      Serial.println("Gyr Not Available");      /* send „GYR Not Available"             */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Gyr.readX();");                /* print Examples name                                */
      if(Gyr.testConnection() == 0){            /* if gyroscope not Available                    */
         Lcd.locate(1, 0);                      /* set LCD print location to 1, 0           */
         Lcd.print("Not Available");            /* print „Gyroscope not Available"               */
      }
   }
}

void loop(void){                                /* Arduino loop                                  */

   if(Gyr.testConnection()){                    /* if gyroscope Available                        */
      char buf[32];
      int x = Gyr.readX();                      /* read X axis of gyroscope                         */

      sprintf(buf, "X =%6d", x);
      Serial.println(buf);                      /* send X axis of gyroscope to UART0                */

      if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
         Lcd.locate(1, 0);                      /* set LCD print location to 1, 0           */
         Lcd.print(buf);                        /* print X axis of gyroscope on LCD              */
      }
   }
   delay(300);                                  /* wait 300 ms                                   */
}
```

```
unsigned long Gyr.readCounterX(void);
unsigned long Gyr.readCounterY(void);
unsigned long Gyr.readCounterZ(void);
```

Function reads number of data for X,Y or Z axis that passed hardware filtering in gyroscope and software filtering according to the parameters obtained by calibration. If hardware and software filtering of gyroscope is well adjusted, an increase the number of data that passed mentioned filtering shouled be ignored while gyroscope is static.

Function prototype:

    emoro_gyr.h

Arguments:

    None

Return value:

```
unsigned long – Result of function: number of angular data rates that have passed filtering
      (0)   –  Error: Gyroscope not in Advanced mode / none data passed filtering
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example za X os:

```
void setup(void){                               /* Arduino setup                                 */
   Serial.begin(9600);                          /* initialize UART 9600 bps                 */
   Serial.println("Example: Gyr.init();");      /* send Examples name                       */
   if(ReadEmoroHardware() & GYR_AVAILABLE)     /* if gyroscope is initialized                    */
      Serial.println("Gyr Available");         /* send „GYR Available“                 */
   else
      Serial.println("Gyr Not Available");     /* send „GYR Not Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
         Gyr.init();                           /* initialization of gyroscope in Advanced mode with */
                                               /* Standard settings                             */
      Lcd.print("Gyr.init();");                /* print Examples name                           */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0                */
      if(ReadEmoroHardware() & GYR_AVAILABLE){ /* if gyroscope is available                     */
         Lcd.print("Available");               /* print „Gyroscope Available“                   */
      }
      else
         Lcd.print("Not Available");           /* print „Gyroscope Not Available“               */
   }
}

void loop(void){                               /* Arduino loop                                  */
   int res;                                    /* return value of function                      */
   double x_deg, y_deg, z_deg;                 /* angular position for each axis                  */
   char buf[64];                               /* additional buffer for printing                  */
   unsigned long xCounter;                     /* variable to store the number of data that have passed */
                                               /* filtering                                     */
   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position of all 3 axes      */

   if(res == 0){                               /* if it is successfully readed print message */
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
      xCounter=Gyr.readCounterX();             /* storing number of data that have passed    */
                                               /* filtering                                     */
      sprintf(buf, "Gyr.readCounterX()=%u", xCounter); /* print values              */
      Serial.println(buf);
   }
  else
    Serial.println("Can't read angular position."); /* message for unseccessfully reading of angular */
                                                   /*position */

  delay(300);                                  /* wait 300 ms                        */
}
```

```
unsigned char Gyr.Dready(void);
```

Function returns state of DRDY signal from gyroscope. For details of DRDY signal functionality use datasheet for the gyroscope L3GD20.

Function prototype:
emoro_gyr.h

Arguments:
None

Return value:
unsigned char – Result of function: state of DRDY signal: 0 or 1

Supported moduls:
EMoRo 2560, Extend board – EMoRo GLAM

```
int Gyr.Calibration(unsigned int samples);
```

Function performs adjustment of software filtering of gyroscope used in Advanced mode. Argument is number of samples which were used for calculation of data value for software filtering. Function calculates dc_offset and noise for each axis. While calibration, gyro should be without movement. During initialization of the gyroscope in Advanced mode, this calibration is performed automatically with predefined number of sampling (100). Afterwards execution of calibration can improve filtering and thus calculation of angular position of gyroscope.

Function prototype:

```
      emoro_gyr.h
```

Arguments:

```
      unsigned int samples - number of samples which were used for calculation of dc_offset and noise
```

Return value:

```
      int – Result of function:
              (0)     – successfully calibration
              (-1)    – Error: gyroscope is not in Advanced mode
              (-2)    – Error: calibration failed
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                 /* Arduino setup                                       */
   Serial.begin(9600);                            /* initialize UART 9600 bps                       */
   Serial.println("Example: Gyr.init();");        /* send Examples name                             */
   if(ReadEmoroHardware() & GYR_AVAILABLE)        /* if gyroscope is initialized                         */
      Serial.println("Gyr Available");            /* send „GYR Available“                      */
   else
      Serial.println("Gyr Not Available");        /* send „GYR Not Available“                    */

   if(ReadEmoroHardware() & LCD_AVAILABLE){       /* if LCD on EMoRo GLAM is initialized   */
         Gyr.init();                              /* initialization of gyroscope in Advanced mode with */
                                                  /* Standard settings                                */
      Lcd.print("Gyr.init();");                   /* print Examples name                              */
      Lcd.locate(1, 0);                           /* set LCD print location to 1, 0              */
      if(ReadEmoroHardware() & GYR_AVAILABLE){    /* if gyroscope is available                           */
         Lcd.print("Available");                  /* print „Gyroscope Available“                         */
      }
      else
         Lcd.print("Not Available");              /* print „Gyroscope Not Available“                     */
   }
}

void loop(void){                                  /* Arduino loop                                        */
   int res;                                       /* return value of function                         */
   double x_deg, y_deg, z_deg;                    /* angular position for each axis                         */
   char buf[64];                                  /* additional buffer for printing                         */

   res = Gyr.readDegrees(&x_deg, &y_deg, &z_deg); /* reading angular position of all 3 axes       */

   if(res == 0){                                  /* if it is successfully readed print message */
      sprintf(buf, "Current position: X =%3d, Y =%3d, Z =%3d", (int)x_deg, (int)y_deg, (int)z_deg);
      Serial.println(buf);
   }
  else
    Serial.println("Can't read angular position."); /* message for unseccessfully reading of angular */
                                                      /*position */
  if( (ReadEmoroHardware() & LCD_AVAILABLE) && (res==0) ){
      Lcd.locate(0, 0);                           /* set LCD print location to 0, 0     */
      Lcd.print(" X    Y    Z");                  /* print axes names                               */
      Lcd.locate(1, 0);                           /* set LCD print location to 1, 0   */
      sprintf(buf, "%3d %3d %3d", (int)x_deg, (int)y_deg, (int)z_deg);  /* print kutnu poziciju      */
      Lcd.print(buf);
  }
  else if(ReadEmoroHardware() & LCD_AVAILABLE){ /*message for unseccessfully reading of angular position*/
      Lcd.locate(1, 0);
      Lcd.print("Can't read      ");
  }
  if(ReadEmoroHardware() & SW_AVAILABLE){      /* if pushbuttons are available                     */
     if(ReadSwitch(SW_1)){                      /* by pressing the SW1 current position of gyroscope  */
        Serial.println("Reset current position. X=0, Y=0, Z=0.");  /* is setted to 0 on all 3 axes      */
        Gyr.resetDegrees();
     }
     else if(ReadSwitch(SW_2)){                 /* by pressing the SW2 pushbutton gyroscope in Advanced  */
        Gyr.stop();                             /* mode stops and thus save microcontrollers time for    */
        Serial.println("Terminate Gyro Advanced mode."); /* data and interrupt processing */
        }
     else if(ReadSwitch(SW_3)){              /* by pressing the SW3 pushbutton starts calibration with */
        Gyr.Calibration(300);                /* 300 samples                          */
        Serial.println("Calibration has finished.");        }
  }
  delay(300);                                 /* wait 300 ms                          */
}
```

```
int Gyr.dc_offsetX(void);
int Gyr.dc_offsetY(void);
int Gyr.dc_offsetZ(void);
```

Function reads dc_offset parameter obtained by calibration of gyroscope in static condition for axis X, Y or Z. The parameter dc_offset is used for software filtering data obtained from gyroscope.

Function prototype:
emoro_gyr.h

Arguments:
None

Return value:
unsigned char – Result of function: dc_offset of readed axes in static condition

Supported moduls:
EMoRo 2560, Extend board – EMoRo GLAM

```
int Gyr.noiseX(void);
int Gyr.noiseY(void);
int Gyr.noiseZ(void);
```

Function reads noise parameter obtained by calibration of gyroscope in static condition for axis X, Y or Z. The parameter noise is used for software filtering data obtained from gyroscope.

Function prototype:
emoro_gyr.h

Arguments:
None

Return value:
int – Result of function: noise of readed axes in static condition

Supported moduls:
EMoRo 2560, Extend board – EMoRo GLAM

```
int Mag.init(void);
```

Function initialize Compass (Magnetometer) on EMoRo GLAM module. Before **setup();** and **loop();** Arduino IDE will initialize Compass so re-initialization is not required.

Function prototype:

    emoro_mag.h

Arguments:

    None

Return value:

```
int – Result of function:
        (0)   –  Compass successfully initialized
        (-1)  –  Error: Unsuccessful communication with compass
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                             /* Arduino setup                                  */

  /* Compass is initialized so re-initialization is not required                      */

  Serial.begin(9600);                         /* initialize UART 9600 bps                    */
  Serial.println("Example: Mag.init();");     /* send Examples name                          */
  if(ReadEmoroHardware() & MAG_AVAILABLE)     /* if compass is initialized                      */
     Serial.println("Mag Available");         /* send „MAG Available“                    */
  else
     Serial.println("Mag Not Available");     /* send „MAG Not Available“                 */

  if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
     Lcd.print("Mag.init();");                /* print Examples name                            */
     Lcd.locate(1, 0);                        /* set LCD print location to 1, 0           */
     if(ReadEmoroHardware() & MAG_AVAILABLE)  /* if compass is Available                          */
        Lcd.print("Available");               /* print „Compass Available“                     */
     else
        Lcd.print("Not Available");           /* print „Compass Not Available“                  */
  }
}
void loop(void){                              /* Arduino loop                                   */
}
```

```
unsigned char Mag.testConnection(void);
```

Function test availability of compass (magnetometer) on EMoRo GLAM module.

Function prototype:

    emoro_mag.h

Arguments:

    None

Return value:

```
unsigned char    –  Result of function:
                    (0)   –  Compass not Available
                    (1)   –  Compass Available
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                /* Arduino setup                                 */

   Serial.begin(9600);                                   /* initialize UART 9600 bps            */
   Serial.println("Example: Mag.testConnection();");    /* send Examples name                  */
   if(Mag.testConnection())                      /* if compass is Available                       */
      Serial.println("Mag Available");           /* send „MAG Available“                  */
   else
      Serial.println("Mag Not Available");       /* send „MAG Not Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Mag.connection()");             /* print Examples name                         */
      Lcd.locate(1, 0);                          /* set LCD print location to 1, 0           */
      if(Mag.testConnection())                   /* if Compass is Available                       */
         Lcd.print("Available");                 /* print „Compass Available“                   */
      else
         Lcd.print("Not Available");             /* print „Compass Not Available“                 */
   }
}

void loop(void){                                 /* Arduino loop                                  */
}
```

```
int Mag.read(int *direction, int *inclination, int *strenght);
```

Function reads direction, inclination and strength of magnetic field from compass on EMoRo GLAM module. Before reading of Compass it is necessary to calibrate (push SW_2 pushbutton during reset of EMoRo 2560 controller). Function update inclination and direction in degrees, strenght of magnetic field in uT.

Function prototype:
```
      emoro_mag.h
```

Arguments:
```
      int *direction     – a pointer to the variable for storing compass direction
      int *inclination   – a pointer to the variable for storing compass inclination
      int *strength      – a pointer to the variable for storing compass strength
```

Return value:
```
      int – Result of function:
               (0)   –  Compass successfully readed
               (-1)  –  Error: Unsuccessful communication with Compass
```

Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                /* Arduino setup                                 */

   Serial.begin(9600);                           /* initialize UART 9600 bps                */
   Serial.println("Example: Mag.read();");       /* send Examples name                      */
   if(Mag.testConnection() == 0)                 /* if Compass not Available                      */
      Serial.println("Mag Not Available");       /* send „MAG Not Available“                 */

   if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Mag.read();");                  /* print Examples name                         */
      if(Mag.testConnection() == 0){             /* if Compass not Available                      */
         Lcd.locate(1, 0);                       /* set LCD print location to 1, 0           */
         Lcd.print("Not Available");             /* print „Compass Not Available“                 */
      }
   }
}

void loop(void){                                 /* Arduino loop                                          */
   int direction, inclination, strength, res;

   res = Mag.read(&direction, &inclination, &strength);

   if(res == 0){                                 /* if compass is readed                              */
      char buf[64];                              /* send compass data to UART0                     */
      sprintf(buf, "Dir =%4d, Inc =%4d, Str =%4d", direction, inclination, strength);
      Serial.println(buf);

      if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
         Lcd.locate(1, 0);                       /* set LCD print location to 1, 0              */
         sprintf(buf,"Dir =%4d", direction);     /* print direction of compass on LCD              */
         Lcd.print(buf);
      }
   }
   delay(300);                                   /* wait 300 ms                                           */
}
```

```
int Mag.readDirection(void);
```

Function reads direction of compass on EMoRo GLAM module. Before reading of compass it is necessary to calibrate (push SW_2 pushbutton during reset of EMoRo 2560 controller). Function returns direction in degrees.

Function prototype:
      emoro_mag.h

Arguments:
      None

Return value:
      int   – Direction of Compass

Supported moduls:
      EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                /* Arduino setup                                         */

   Serial.begin(9600);                           /* initialize UART 9600 bps                       */
   Serial.println("Example: Mag.readDirection();");    /* send Examples name                       */
   if(Mag.testConnection() == 0)                 /* if Compass not Available                           */
      Serial.println("Mag Not Available");       /* send „MAG Not Available“                    */

   if(ReadEmoroHardware() & LCD_AVAILABLE){      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Direction();");                 /* print Examples name                               */
      if(Mag.testConnection() == 0){             /* if Compass not Available                           */
         Lcd.locate(1, 0);                       /* set LCD print location to 1, 0              */
         Lcd.print("Not Available");             /* print „Compass not Available“                      */
      }
   }
}

void loop(void){                                 /* Arduino loop                                          */

   if(Mag.testConnection()){                     /* if Compass is Available                            */
      char buf[32];
```

```
    int direction = Mag.readDirection();      /* read Compass direction                         */
    sprintf(buf, "Dir =%4d", direction);
    Serial.println(buf);                      /* send direction to UART0                */

    if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(1, 0);                       /* set LCD print location to 1, 0         */
      Lcd.print(buf);                         /* print direction on LCD                   */
    }
  }
  delay(300);                                 /* wait 300 ms                                    */
}
```

```
int Mag.readInclination(void);
```

Function reads inclination of compass on EMoRo GLAM module. Before reading of compass it is necessary to calibrate (push SW_2 pushbutton during reset of EMoRo 2560 controller). Function returns inclination in degrees.

Function prototype:
    emoro_mag.h

Arguments:
    None

Return value:
    int   – Inclination of Compass

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                             /* Arduino setup                                  */

  Serial.begin(9600);                         /* initialize UART 9600 bps                 */
  Serial.println("Example: Mag.readInclination();");    /* send Examples name             */
  if(Mag.testConnection() == 0)               /* if Compass not Available                       */
    Serial.println("Mag Not Available");      /* send „MAG Not Available“               */

  if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("Inclination();");              /* print Examples name                          */
    if(Mag.testConnection() == 0){            /* if Compass not Available                       */
      Lcd.locate(1, 0);                       /* set LCD print location to 1, 0         */
      Lcd.print("Not Available");             /* print „Compass not Available“                  */
    }
  }
}
void loop(void){                              /* Arduino loop                                   */

  if(Mag.testConnection()){                   /* if Compass is Available                        */
    char buf[32];
    int inclination = Mag.readInclination(); /* read Compass inclination                       */
    sprintf(buf, "Inc =%4d", inclination);
    Serial.println(buf);                      /* send inclination to UART0                */

    if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(1, 0);                       /* set LCD print location to 1, 0         */
      Lcd.print(buf);                         /* print inclination on LCD                 */
    }
  }
  delay(300);                                 /* wait 300 ms                                    */
}
```

```
int Mag.readStrength(void);
```

Function reads strength of magnetic field. Before reading of compass it is necessary to calibrate (push SW_2 pushbutton during reset of EMoRo 2560 controller). Function returns strength in uT.

Function prototype:

```
      emoro_mag.h
```

Arguments:

```
      None
```

Return value:

```
      int   - Strength of magnetic field
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                                  /* Arduino setup                                         */

   Serial.begin(9600);                             /* initialize UART 9600 bps                         */
   Serial.println("Example: Mag.readStrength();");      /* send Examples name                        */
   if(Mag.testConnection() == 0)                   /* if Compass not Available                              */
      Serial.println("Mag Not Available");         /* send „MAG Not Available“                       */

   if(ReadEmoroHardware() & LCD_AVAILABLE){        /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("Strength();");                    /* print Examples name                                */
      if(Mag.testConnection() == 0){               /* if Compass not Available                              */
         Lcd.locate(1, 0);                         /* set LCD print location to 1, 0                 */
         Lcd.print("Not Available");               /* print „Compass not Available“                         */
      }
   }
}

void loop(void){                                   /* Arduino loop                                          */

   if(Mag.testConnection()){                       /* if Compass is Available                               */
      char buf[32];
      int strength = Mag.readStrength();           /* read strength of magnetic field                       */

      sprintf(buf, "Str =%4d", strength);
      Serial.println(buf);                         /* send strength of magnetic field to UART0              */

      if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized   */
         Lcd.locate(1, 0);                         /* set LCD print location to 1, 0                 */
         Lcd.print(buf);                           /* print strength of magnetic field on LCD               */
      }
   }
   delay(300);                                     /* wait 300 ms                                           */
}
```

```
int Mag.calibrate(unsigned char step);
```

Function calibrates compass and before first reading of compass it is necessary to calibrate (push SW_2 pushbutton during reset of EMoRo 2560 controller).
The values of calibrated compass are stored in EEPROM memory and loaded during the initialization of EMoRo 2560 controller.
The calibration function is necessary to call 3 times using the argument = step 1, step 2 and step = 3. Before calling the function it is necessary to set the controller in the following states:

1. Place controller on a flat surface and call **Mag.calibrate(1);**
2. Rotate controller for 180 degrees and call **Mag.calibrate(2);**
3. Turn controller so that LCD is on a flat surface and call **Mag.calibrate(3);**

Compass calibration can be done during the initialization of the controller by holding down push button SW_2. After launching of calibration program is necessary to follow the instructions printed on the LCD.

Function prototype:
```
      emoro_mag.h
```

Arguments:
```
      unsigned char step -  step of compass calibration [1-3]
```

Return value:
```
      int – Result of function:
              (0)   –  Step of compass calibration successfully executed
              (-1)  –  Error: Unsuccessful communication with compass
              (-2)  –  Error: Step of compass calibration not in range [1-3]
```

Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                               /* Arduino setup                                       */

   /* Push and hold the SW_2 pushbutton and reset controller to start program for compass calibration.  */

   Serial.begin(9600);                          /* initialize UART 9600 bps                      */
   Serial.println("Example: Mag.calibrate();");           /* send Examples name                  */
   if(Mag.testConnection() == 0)                /* if Compass not Available                            */
      Serial.println("Mag Not Available");      /* send „MAG Not Available“                     */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("calibrate();");                /* print Examples name                              */
      if(Mag.testConnection() == 0){            /* if Compass not Available                            */
         Lcd.locate(1, 0);                      /* set LCD print location to 1, 0               */
         Lcd.print("Not Available");            /* print „Compass not Available“                       */
      }
   }
}

void loop(void){                                /* Arduino loop                                        */
}
```

```
void InitSwitch(void);
```

Function initialize pushbuttons SW_1 – SW_4 on EMoRo GLAM module. Before **setup();** and **loop();** Arduino IDE will initialize pushbuttons so re-initialization is not required.

Function prototype:
    emoro_switch.h

Arguments:
    None

Return value:
    None

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:
```
void setup(void){                               /* Arduino setup                                    */

   /* Pushbuttons are initialized so re-initialization is not required                         */

   Serial.begin(9600);                          /* initialize UART 9600 bps                      */
   Serial.println("Example: InitSwitch();");    /* send Examples name                            */
   if(ReadEmoroHardware() & SW_AVAILABLE)       /* if pushbuttons SW_1 – SW_4 are initialized          */
      Serial.println("Sw Available");           /* send „Pushbuttons Available“                  */
   else
      Serial.println("Sw Not Available");       /* send „Pushbuttons Not Available“              */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("InitSwitch();");               /* print Examples name                             */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0              */
      if(ReadEmoroHardware() & SW_AVAILABLE)    /* if pushbuttons are available                          */
         Lcd.print("Available");                /* print „Available“                           */
      else
         Lcd.print("Not Available");            /* print „Not Available“                    */
   }
}

void loop(void){                                /* Arduino loop                                     */
}
```

```
unsigned char ReadSwitch(unsigned char sw);
```

Function reads state of pushbutton SW_1 – SW_4 on EMoRo GLAM module.

Function prototype:
    emoro_switch.h

Arguments:
```
    unsigned char sw   –  Pushbutton SW_1 – SW_4
```

Return value:
```
    unsigned char      –  Result of function:
                          (0)  –  Pushbutton is active
                          (1)  –  Pushbutton is not active
```

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                              /* Arduino setup                                    */

   Serial.begin(9600);                         /* initialize UART 9600 bps                    */
   Serial.println("Example: ReadSwitch();");   /* send Examples name                          */
   if(ReadEmoroHardware() & SW_AVAILABLE)      /* if pushbuttons SW_1 – SW_4 are initialized        */
      Serial.println("Sw Available");          /* send „Pushbuttons Available“                */
   else
      Serial.println("Sw Not Available");      /* send „Pushbuttons Not Available“            */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("ReadSwitch();");              /* print Examples name                           */
   }
}

void loop(void){                                  /* Arduino loop                                  */
   static unsigned char last_sw;
   unsigned char temp_sw=0;

   if(ReadEmoroHardware() & SW_AVAILABLE){        /* if pushbuttons are available                      */

      temp_sw |= (ReadSwitch(SW_1) << 0);         /* read state of pushbuttons in temp_sw variable      */
      temp_sw |= (ReadSwitch(SW_2) << 1);
      temp_sw |= (ReadSwitch(SW_3) << 2);
      temp_sw |= (ReadSwitch(SW_4) << 3);

      if(temp_sw != last_sw){                     /* if je stanje tipkala promjenjeno             */
         char buf[16];
         last_sw = temp_sw;                       /* save new state of pushbuttons in last_sw variable  */
         sprintf(buf, "Sw = %2d", 15-temp_sw);    /* sort pushbuttons 0 – 15 and print them in buf    */
         Serial.println(buf);                     /* print state of pushbutton to UART 0                */
         if(ReadEmoroHardware() & LCD_AVAILABLE){ /* if LCD on EMoRo GLAM is initialized*/
            Lcd.locate(1, 0);                     /* set LCD print location to 1, 0           */
            Lcd.print(buf);                       /* print state of pushbutton on LCD                   */
         }
      }
   }
}
```

```
int Bluetooth.init(void);
```

Function initialize Bluetooth on EMoRo GLAM module. Before **setup();** and **loop();** Arduino IDE will initialize bluetooth and Serial1 bus so re-initialization with **Bluetooth.init();** and **Serial1.begin(38400);** is not required.Bluetooth is initialized via UART 1 bus and for reading and sending data uses **Serial1** functions. In the case that during the initialization of EMoRo 2560 controller SW_1 pushbutton is pressed, bluetooth module will load default settings (Name: EMoRo 2560, Passkey: 0000).

Function prototype:

    emoro_bluetooth.h

Arguments:

    None

Return value:

```
int  – Result of function:
          (0)      - Bluetooth successfully initialized
          (-1)     - Error: Unsuccessful communication with Bluetooth module
```

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                         /* Arduino setup                                */

  /* Bluetooth and Serial1 are initialized so re-initialization is not required       */

  Serial.begin(9600);                     /* initialize UART 9600 bps                 */
  Serial.println("Example: Bluetooth.init();");    /* send Examples name              */
  if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)    /* if bluetooth is initialized       */
    Serial.println("BT Available");       /* send „Bluetooth Available“            */
  else
    Serial.println("BT Not Available");   /* send „Bluetooth Not Available“          */

                  if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("Bluetooth.init()");        /* print Examples name                          */
    Lcd.locate(1, 0);                     /* set LCD print location to 1, 0            */
    if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)    /* if Bluetooth is Available           */
      Lcd.print("Available");             /* print „Bluetooth Available“                */
    else
      Lcd.print("Not Available");         /* print „Bluetooth Not Available“            */
  }
}

void loop(void){                          /* Arduino loop                                 */
  if(Bluetooth.connection()){             /* if Bluetooth is connected                    */
    Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth       */
  }
  delay(500);                             /* wait 500 ms                                  */
}
```

```
unsigned char Bluetooth.testConnection(void);
```

Function test availability of Bluetooth on EMoRo GLAM module.

Function prototype:

    emoro_bluetooth.h

Arguments:

    None

Return value:
```
      unsigned char    –  Result of function:
                          (0)   –  Bluetooth not Available
                          (1)   –  Bluetooth Available
```

Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                               /* Arduino setup                                    */

   Serial.begin(9600);                                       /* initialize UART 9600 bps         */
   Serial.println("Example: Bluetooth.testConnection();");   /* send Examples name               */
   if(Bluetooth.testConnection())               /* if bluetooth is Available                        */
      Serial.println("BT Available");           /* send „Bluetooth Available“              */
   else
      Serial.println("BT Not Available");       /* send „Bluetooth Not Available“       */

   if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("testConnection()");            /* print Examples name                         */
      Lcd.locate(1, 0);                         /* set LCD print location to 1, 0          */
      if(Bluetooth.testConnection())            /* if bluetooth is Available                        */
         Lcd.print("Available");                /* print „Bluetooth Available“                 */
      else
         Lcd.print("Not Available");            /* print „Bluetooth Not Available“                  */
   }
}

void loop(void){                                /* Arduino loop                                     */
   if(Bluetooth.connection()){                  /* if bluetooth is connected                           */
      Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth        */
   }
   delay(500);                                  /* wait 500 ms                                      */
}
```

```
int Bluetooth.changeName(char *new_name);
```

Function changes name of bluetooth module.

Function prototype:
```
      emoro_bluetooth.h
```

Arguments:
```
      char* new_name   - Pointer to a new bluetooth name
```

Return value:
```
      int   -  (0)   – Bluetooth name successfully changed
               (-1)  – Error: Bluetooth module does not support characters of a new name
```

Supported moduls:
```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                               /* Arduino setup                                    */
   int res;

   Serial.begin(9600);                                       /* initialize UART 9600 bps         */
   Serial.println("Example: Bluetooth.changeName();");       /* send Examples name               */
   if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)             /* if bluetooth is Available        */
      Serial.println("BT Available");           /* send „Bluetooth Available“              */
   else
      Serial.println("BT Not Available");      /* send „Bluetooth Not Available“          */

   res = Bluetooth.changeName("Arduino BT");   /* change name of bluetooth module              */

   if(res == 0)                                /* if Bluetooth name successfully changed       */
      Serial.println("Successfully");          /* send „Name successfully changed“        */
   else
      Serial.println("Failed");                /* send „Ime nije promjenjeno“             */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("changeName();");              /* print Examples name                          */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0            */
      if(res == 0)
         Lcd.print("Successfully");            /* print message „Name successfully changed“          */
      else
         Lcd.print("Failed");                  /* print message „Name not changed“          */
   }
}

void loop(void){                               /* Arduino loop                                  */
   if(Bluetooth.connection()){                 /* if bluetooth is connected                        */
      Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth         */
   }
   delay(500);                                 /* wait 500 ms                                   */
}
```

```
int Bluetooth.changePasskey(char *new_passkey);
```

Function changes passkey of bluetooth module.

Function prototype:

```
      emoro_bluetooth.h
```

Arguments:

```
      char* new_passkey  - Pointer to a new passkey
```

Return value:

```
      int  -  (0)   – Bluetooth passkey successfully changed
              (-1)  – Error: Bluetooth module does not support characters of a new passkey
```

Supported moduls:

```
      EMoRo 2560, Extend board – EMoRo GLAM
```

Example:

```
void setup(void){                              /* Arduino setup                                 */
   int res;

   Serial.begin(9600);                                   /* initialize UART 9600 bps        */
   Serial.println("Example: Bluetooth.changePasskey();"); /* send Examples name              */
   if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)         /* if bluetooth is Available            */
      Serial.println("BT Available");          /* send „Bluetooth Available“               */
   else
      Serial.println("BT Not Available");      /* send „Bluetooth Not Available“          */

   res=Bluetooth.changePasskey("1234");        /* change passkey of bluetooth module           */

   if(res == 0)                                /* if bluetooth passkey successfully changed       */
      Serial.println("Successfully");          /* send „Passkey successfully changed“     */
   else
```

```
    Serial.println("Failed");                /* send „Passkey not changed“         */

  if(ReadEmoroHardware() & LCD_AVAILABLE){   /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("changePasskey();");           /* print Examples name                           */
    Lcd.locate(1, 0);                        /* set LCD print location to 1, 0            */
    if(res == 0)
      Lcd.print("Successfully");             /* print message „Passkey successfully changed“       */
    else
      Lcd.print("Failed");                   /* print message „Passkey not changed“          */
  }
}

void loop(void){                             /* Arduino loop                                   */
  if(Bluetooth.connection()){                /* if bluetooth is connected                      */
    Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth            */
  }
  delay(500);                                /* wait 500 ms                                    */
}
```

```
int Bluetooth.changeNameAndPasskey(char *new_name, char *new_passkey);
```

Function changes name and passkey of bluetooth module.

Function prototype:
```
    emoro_bluetooth.h
```

Arguments:
```
    char* new_name     - Pointer to a new name
    char* new_passkey  - Pointer to a new passkey
```

Return value:
```
    int  -  (0)   – Bluetooth passkey and name successfully changed
            (-1)  – Error: Bluetooth module does not support characters of a new passkey or a new name
```

Supported moduls:
```
    EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                            /* Arduino setup                                  */
  int res;

  Serial.begin(9600);                                             /* initialize UART 9600 bps*/
  Serial.println("Example: Bluetooth.changeNameAndPasskey();");   /* send Examples name      */
  if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)                   /* if bluetooth is Available     */
    Serial.println("BT Available");          /* send „Bluetooth Available“              */
  else
    Serial.println("BT Not Available");      /* send „Bluetooth Not Available“          */

  /* change name and passkey of bluetooth module                                               */
  res = Bluetooth.changeNameAndPasskey("Arduino BT", "1234");

  if(res == 0)                               /* if name and passkey successfully changed        */
    Serial.println("Successfully");          /* send „Name and passkey changed“          */
  else
    Serial.println("Failed");                /* send „Name and passkey not changed“    */

  if(ReadEmoroHardware() & LCD_AVAILABLE){   /* if LCD on EMoRo GLAM is initialized   */
    Lcd.print("NameAndPasskey()");           /* print Examples name                           */
    Lcd.locate(1, 0);                        /* set LCD print location to 1, 0            */
    if(res == 0)
```

```
        Lcd.print("Successfully");             /* print message „Name and passkey changed“        */
     else
        Lcd.print("Failed");                   /* print message „Name and passkey not changed“        */
   }
}

void loop(void){                               /* Arduino loop                                        */
   if(Bluetooth.connection()){                 /* if bluetooth is connected                             */
      Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth               */
   }
   delay(500);                                 /* wait 500 ms                                         */
}
```

```
char *Bluetooth.readName(void);
```

Functio reads name of bluetooth module.

Function prototype:

    emoro_bluetooth.h

Arguments:

    None

Return value:

    char* - Pointer to a name of bluetooth module

Supported moduls:

    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                              /* Arduino setup                                       */

   Serial.begin(9600);                                 /* initialize UART 9600 bps               */
   Serial.println("Example: Bluetooth.readName();");/* send Examples name                       */
   if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE){    /* if bluetooth is Available                     */
      Serial.println("BT Available");                /* send „Bluetooth Available“            */
      Serial.println(Bluetooth.readName());          /* send name of bluetooth module                 */
   }
   else
      Serial.println("BT Not Available");            /* send „Bluetooth Not Available“     */

   if(ReadEmoroHardware() & LCD_AVAILABLE){          /* if LCD on EMoRo GLAM is initialized*/
      Lcd.print("readName();");                      /* print Examples name                      */
      Lcd.locate(1, 0);                              /* set LCD print location to 1, 0        */
      if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)  /* if bluetooth is Available                     */
         Lcd.print(Bluetooth.readName());            /* print name of bluetooth module                */
      else
         Lcd.print("Not Available");                 /* print message „Bluetooth Not Available“       */
   }
}

void loop(void){                               /* Arduino loop                                        */
   if(Bluetooth.connection()){                 /* if bluetooth is connected                        */
      Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth               */
   }
   delay(500);                                 /* wait 500 ms                                         */
}
```

```
char *Bluetooth.readPasskey(void);
```

Functio reads passkey of bluetooth module.

Function prototype:
    emoro_bluetooth.h

Arguments:
    None

Return value:
    char* - Pointer to a passkey of bluetooth module

Supported moduls:
    EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                          /* Arduino setup                          */

   Serial.begin(9600);                                     /* initialize UART 9600 bps           */
   Serial.println("Example: Bluetooth.readPasskey();");   /* send Examples name                 */
   if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE){          /* if bluetooth is Available              */
      Serial.println("BT Available");                      /* send „Bluetooth Available“      */
      Serial.println(Bluetooth.readPasskey());             /* send passkey of bluetooth module     */
   }
   else
      Serial.println("BT Not Available");             /* send „Bluetooth nije Available“    */

   if(ReadEmoroHardware() & LCD_AVAILABLE){           /* if LCD on EMoRo GLAM is initialized*/
      Lcd.print("readPasskey();");                    /* print Examples name                     */
      Lcd.locate(1, 0);                               /* set LCD print location to 1, 0       */
      if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)   /* if bluetooth is Available                */
         Lcd.print(Bluetooth.readPasskey());          /* print passkey of bluetooth module        */
      else
         Lcd.print("Not Available");                  /* print message „Bluetooth Not Available“      */
   }
}

void loop(void){                                 /* Arduino loop                                  */
   if(Bluetooth.connection()){                   /* if bluetooth is connected                   */
      Serial1.println("Hello from Bluetooth"); /* send Hello message via bluetooth          */
   }
   delay(500);                                   /* wait 500 ms                                   */
}
```

```
unsigned char Bluetooth.connection(void);
```

Functio reads state of bluetooth connection (Bluetooth connected, Bluetooth not connected).

Function prototype:
    emoro_bluetooth.h

Arguments:
    None

Return value:
```
    unsigned char -  (1)   – Bluetooth connected
                     (0)   – Bluetooth not connected
```

Supported moduls:

EMoRo 2560, Extend board – EMoRo GLAM

Example:

```
void setup(void){                                         /* Arduino setup                        */

   Serial.begin(9600);                                    /* initialize UART 9600 bps         */
   Serial.println("Example: Bluetooth.connection();");    /* send Examples name               */
   if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)          /* if bluetooth is Available            */
      Serial.println("BT Available");                     /* send „Bluetooth Available“      */
   else
      Serial.println("BT Not Available");                 /* send „Bluetooth Not Available“ */

   if(ReadEmoroHardware() & LCD_AVAILABLE){          /* if LCD on EMoRo GLAM is initialized*/
      Lcd.print("BT.connection();");                 /* print Examples name                     */
      Lcd.locate(1, 0);                              /* set LCD print location to 1, 0         */
      if(ReadEmoroHardware() & BLUETOOTH_AVAILABLE)  /* if bluetooth is Available                 */
         Lcd.print("Down");                          /* print state of BT connection              */
      else
         Lcd.print("Not Available");                 /* print message „Bluetooth Not Available“       */
   }
}

void loop(void){                                   /* Arduino loop                                */
   static unsigned char last_bt_state = 0;

   if(Bluetooth.connection() != last_bt_state){   /* if state of BT cennection is changed  */
      last_bt_state = Bluetooth.connection();
      if(last_bt_state)                            /* send status of BT connection to UART 0           */
         Serial.println("Bluetooth UP");
      else
         Serial.println("Bluetooth Down");

      if(ReadEmoroHardware() & LCD_AVAILABLE){     /* if LCD on EMoRo GLAM is initialized*/
         Lcd.locate(1, 0);                         /* set LCD print location to 1, 0         */
         if(last_bt_state)                         /* print state of BT connection  on LCD        */
            Lcd.print("UP  ");
         else
            Lcd.print("Down");
      }
   }
   if(Bluetooth.connection()){                     /* if bluetooth is connected                    */
      Serial1.println("Hello from Bluetooth");     /* send Hello message via bluetooth         */
   }
   delay(500);                                     /* wait 500 ms                                 */
}
```

```
int DS1820.attach(unsigned char port);
```

Function initialize temperature sensor DS1820 on ports IO_0 – IO_16, ADC_0 – ADC_7, SERVO_0 – SERVO_7, GPP_0_A – GPP_7_B, PWM_0 – PWM_5, EX_IO_0 – EX_IO_17.

Function prototype:
```
    emoro_ds1820.h
```

Arguments:
```
    unsigned char port -  DS1820 port (IO_0 – IO_16, ADC_0 – ADC_7, SERVO_0 – SERVO_7, GPP_0_A, GPP_7_B,
                                      PWM_0 – PWM_5, EX_IO_0 – EX_IO_17)
```

Return value:
```
    int – Result of function:
        (0)    –  Temperature sensor initialized
        (-1)   –  Error: Port out of range
        (-2)   –  Error: Unsuccessful communication with temperature sensor DS1820
```

Supported moduls:
```
    EMoRo 2560, Extend board – EMoRo GLAM
```

Example:
```
void setup(void){                              /* Arduino setup                                  */

   int res = DS1820.attach(IO_0);              /* initialize DS1820 on port IO_0             */

   Serial.begin(9600);                               /* initialize UART 9600 bps             */
   Serial.println("Example: DS1820.attach();");      /* send Examples name                   */
   if(res == 0)                                /* if DS1820 is initialized                   */
      Serial.println("Successfully");          /* send „DS1820 initialized“                  */
   else
      Serial.println("Failed");                /* send „DS1820 not initialized“           */

   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("DS1820.attach();");           /* print Examples name                           */
      Lcd.locate(1, 0);                        /* set LCD print location to 1, 0          */
      if(res == 0)                             /* if DS1820 is initialized                   */
         Lcd.print("Successfully");            /* print „DS1820 initialized“                 */
      else
         Lcd.print("Failed");                  /* print „DS1820 not initialized“                */
   }
}

void loop(void){                               /* Arduino loop                                   */
}
```

```
float DS1820.read(unsigned char port);
```

Function reads temperature sensor DS1820 on ports IO_0 – IO_16, ADC_0 – ADC_7, SERVO_0 – SERVO_7, GPP_0_A – GPP_7_B, PWM_0 – PWM_5, EX_IO_0 – EX_IO_17. Before reading of temperature sensor DS1820 is necessary to initialize with **DS1820.attach();** function.

Function prototype:
```
    emoro_ds1820.h
```

Arguments:
```
    unsigned char port -  DS1820 port (IO_0 – IO_16, ADC_0 – ADC_7, SERVO_0 – SERVO_7, GPP_0_A, GPP_7_B,
                                      PWM_0 – PWM_5, EX_IO_0 – EX_IO_17)
```

Return value:

```
      int - Result of function:
            (<200)  -  Temperature of DS1820 sensor
            (200)   -  Error: Unsuccessful communication with temperature sensorom DS1820
```

Supported moduls:

```
      EMoRo 2560, Extend board - EMoRo GLAM
```

Example:

```
void setup(void){                               /* Arduino setup                                      */

   int res = DS1820.attach(IO_0);               /* initialize DS1820 on port IO_0                */

   Serial.begin(9600);                               /* initialize UART 9600 bps                 */
   Serial.println("Example: DS1820.read();");        /* send Examples name                       */

   if(ReadEmoroHardware() & LCD_AVAILABLE)      /* if LCD on EMoRo GLAM is initialized   */
      Lcd.print("DS1820.read();");              /* print Examples name                           */
}

void loop(void){                                /* Arduino loop                                       */
   float temp = DS1820.read(IO_0);              /* read temperature sensor from port IO_0             */

   Serial.print("Temp IO_0 = ");                /* send temperature to UART 0                        */
   Serial.println(temp);

                   if(ReadEmoroHardware() & LCD_AVAILABLE){    /* if LCD on EMoRo GLAM is initialized   */
      Lcd.locate(1, 0);
      Lcd.print("Temp: ");                      /* print temperature on LCD                             */
      Lcd.print(temp);
      Lcd.print("  ");
   }
   delay(500);                                  /* wait 500 ms                                        */
}
```