**Serial Monitor**

Add the following code to your project to make use of the serial monitor.

/\*

\* This function initializes the Serial console for reading and writing.

\*

\* Parameters:

\* - None

\*

\* Returns:

\* - Nothing

\*

\* Usage in Setup block

\* initializeConsole();

\*

\*/

void initializeConsole () {

Serial.begin(9600);

}

/\*

\* This function reads a number from the Serial console.

\*

\* Parameters:

\* - None

\*

\* Returns:

\* - The number typed by a used into the serial console.

\*

\* Usage in ther main loop:

\* int myNumber = 0;

\* myNumber = readANumber();

\*

\*/

int readANumber () {

Serial.println("Please enter a number.");

// Keep waiting for the user to type something

while(Serial.available() == 0) {

}

// Read what was typed

int theNumber = Serial.parseInt();

return theNumber;

}

/\*

\* This function reads a string from the Serial console.

\*

\* Parameters:

\* - None

\*

\* Returns:

\* - The string typed by a used into the serial console.

\*

\* Usage in ther main loop:

\* String myCommand = "None";

\* myCommand = readAString();

\*

\*/

String readAString () {

Serial.println("Please enter a string.");

// Keep waiting for the user to type something

while(Serial.available() == 0) {

}

// Read what was typed

String theString = Serial.readString();

return theString;

}

**External LED**

Add the following code to your project to make use of an external LED.

/\*

\* This function initializes and external LED for use.

\*

\* Parameters:

\* - ledPin is the number of the Arduino pin that the external LED is connected to

\*

\* Returns:

\* - Nothing

\*

\* Usage in Setup block

\* int myLedPin = 11; // Note: Change this value for your configuration

\* initializeLED(myLedPin)

\*

\*/

void initializeLED (int ledPin) {

pinMode(ledPin, OUTPUT);

}

/\*

\* This function a LED a specified number of times with a specified

\* delay between blinks.

\*

\* Parameters:

\* - ledPin is the number of the Arduino pin that the external LED is connected to

\* - blinkTimes is the number of times to blink the LED

\* - delayTime is the delay between turning the LED on and off

\* (the amount of time the LED stays lit)

\*

\* Returns:

\* - Nothing

\*

\* Usage in ther main loop:

\* int myLedPin = 11; // Note: Change this value for your configuration

\* int timesToBlink = 3: // Blink 3 times. Change it to what you need

\* int stayLitTime = 500; // Stay lit 0.5 seconds for each blink

\* blinkTheLED(myLedPin, timesToBlink, stayLitTime);

\*

\*/

void blinkTheLED (int ledPin, int blinkTimes, int delayTime) {

for (int i = 1; i <= blinkTimes; i++)

{

digitalWrite(ledPin, HIGH);

delay(delayTime);

digitalWrite(ledPin, LOW);

delay(delayTime);

}

}

**PIR Motion Sensor**

Add the following code to your project to make use of the PIR Motion Sensor.

/\*

\* This function initializes and the PIR Motion Sensor for use.

\*

\* Parameters:

\* - pirPin is the number of the Arduino pin that the external LED is connected to

\*

\* Returns:

\* - Nothing

\*

\* Usage in Setup block

\* int mypirPin = 11; // Note: Change this value for your configuration

\* initializePIR(myLedPin)

\*

\*/

void initializePIR (int pirPin) {

//the time we give the sensor to calibrate (10-60 secs according to the datasheet)

int calibrationTime = 30;

pinMode(pirPin, INPUT);

digitalWrite(pirPin, LOW);

//give the sensor some time to calibrate

for(int i = 0; i < calibrationTime; i++){

delay(1000);

}

delay(50);

}

/\*

\* This function checks the motion sensor to determine if motion has beed detected

\* and it returns the result. It also determines the beginning and end of continuous

\* motion sequences.

\*

\* The sensor's output pin goes to HIGH if motion is present.

\* However, even if motion is present it goes to LOW from time to time,

\* which might give the impression no motion is present.

\* This program deals with this issue by ignoring LOW-phases shorter than a given time,

\* assuming continuous motion is present during these phases.

\*

\* Parameters:

\* - pirPin is the number of Arduino pin that the motion sensor is connected to

\*

\* Returns:

\* - True if motion was detected.

\* - False is motion was not detected.

\*

\* Usage in ther main loop:

\* int pirPin = 11;

\* boolean motionDetected = false;

\* motionDetected = isPirTriggered(pirPin);

\*

\*/

boolean isPirTriggered (int pirPin) {

boolean isTriggered = false;

//the time when the sensor outputs a low impulse

long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low

//before we assume all motion has stopped

long unsigned int pause = 5000;

boolean lockLow = true;

boolean takeLowTime;

if(digitalRead(pirPin) == HIGH){

digitalWrite(ledPin, HIGH); //the led visualizes the sensors output pin state

if(lockLow){

//makes sure we wait for a transition to LOW before any further output is made:

lockLow = false;

// motion was detected

isTriggered = true;

delay(50);

}

takeLowTime = true;

}

if(digitalRead(pirPin) == LOW){

digitalWrite(ledPin, LOW); //the led visualizes the sensors output pin state

if(takeLowTime){

lowIn = millis(); //save the time of the transition from high to LOW

takeLowTime = false; //make sure this is only done at the start of a LOW phase

}

//if the sensor is low for more than the given pause,

//we assume that no more motion is going to happen

if(!lockLow && millis() - lowIn > pause){

//makes sure this block of code is only executed again after

//a new motion sequence has been detected

lockLow = true;

// motion ended

isTriggered = false;

delay(50);

}

}

return isTriggered;

}

**Photocell Light Sensor**

Add the following code to your project to make use of Photocell Sensor.

/\*

\* Initialization.

\*

\* The Photocell is connected to an Analog input pin and

\* does not require initialization.

\*

\*/

/\*

\* This function reads a number from the Photocell.

\* The number returned depends on the light level measured

\* as follows:

\* lowest light level is 200

\* highest light level is 800

\* medium light levels return between 200 to 800

\* The above constraints may be adjusted as needed

\*

\* Parameters:

\* - photocellPin is the number of analog pin that the photocell is connected to

\*

\* Returns:

\* - A number between 200 to 800 corresponding to the light level measured.

\*

\* Usage in ther main loop:

\* int analogPin = 5;

\* int lightLevel = 0;

\* lightLevel = readPhotocell(analogPin);

\*

\*/

int readPhotocell (int photocellPin) {

int photocellValue = 0;

photocellValue = analogRead(photocellPin);

//adjust the constraint depending on lighting environment.

photocellValue = constrain(photocellValue, 200, 800);

return photocellValue;

}

**Ultrasonic Distance Sensor**

Add the following code to your project to make use of a distance sensor.

/\*

\* Initialization.

\*

\* The Ultrasonicdistance sensor uses an object that must

\* be imported into your program.

\*

\* Add the following code to the top of your program to

\* use the Ultrasonic Distance Sensor Object.

\*

\*/

#include <NewPing.h>

#define TRIGGER\_PIN 12 // Arduino pin tied to trigger pin on the ultrasonic sensor.

#define ECHO\_PIN 11 // Arduino pin tied to echo pin on the ultrasonic sensor.

#define MAX\_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.

/\*

\* This function reads a number from the Photocell.

\* The number returned depends on the light level measured

\* as follows:

\* lowest light level is 200

\* highest light level is 800

\* medium light levels return between 200 to 800

\* The above constraints may be adjusted as needed

\*

\* Parameters:

\* - none

\*

\* Returns:

\* - A distance to the object measured in cm.

\*

\* Usage in ther main loop:

\* int measuredDistance = 0;

\* measuredDistance = readUltrasonic();

\*

\*/

int readUltrasonic () {

delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).

int distanceMeasured = (uS / US\_ROUNDTRIP\_CM); // Convert ping time to distance in cm and print result (0 = outside set distance range)

return distanceMeasured;

}