MPPT-Laderegler Monitoring über einen Web Server mit dem ESP32

Harun Dastekin s0551006[1]

Studiengang: Angewandte Informatik, Drahtlose Netzwerke[1]

Harun.Dastekin@Student.HTW-Berlin.de

Zusammenfassung

Dieser Projektbericht behandelt ein Monitoring System für eine kleine Solaranlage mit Batteriespeicher. Ein ESP32 Node MCU misst anhand von Sensoren den Strom und die Spannung des Solarmoduls und überträgt diese an einen grafisch aufbereiteten Web Server.

1 Einleitung

In unserer heutigen Zeit ist ein Zugang zu Stromanschlüssen maSSgeblich entscheidend für den Komfort. Oft liegt aber eine Steckdose nich genau dort wo man sie gerne hätte. Abhilfe sorgen dafür Solaranlagen die im besten Fall im Inselbetrieb arbeiten können. Mit anderen worten ein abgeschlossenes Energiesystem mit Energiezufuhr, Energiespeicher und einer möglichkeit diese Energie zu nutzen. Um den Energiegewinn ideal zu gestalten gibt es mehrere Faktoren die das beeinflussen. Darunter auch das Zusammenspiel zwischen der Energiezufuhr und dem Energiespeicher mit dem Verbrauch. D.h. eine Solarbatterie darf weder überladen noch tiefenentladen werden. Es muss eine Ladekurve geben, die den Ladevorgang reguliert. Um das zu bewerkstelligen benötigt es ein überwachungssystem der flieSSenden Ströme und der anliegenden Spannungen. Dazu wäre ein erster Schritt diese Anlage überwachen zu können und darauf geht diese Projektarbeit ein. Ein ähnliches Projekt habe ich im nachhinein auf dieser Internetseite gefunden(*Solar Panel Data Monitoring using Arduino and LabView* n. d.).

(a) 12V-Batterie

(b) Solar Laderegler

(c) PV Modul

Abbildung 1: Bestandteile der Solaranlage

2 Haupteil

2.1 Bauteile

Die Solaranlage besteht im Grunde genommen aus drei bzw. vier Bestandteilen. Einer 12V Batterie 1a, einem Laderegler 1b und dem Photovoltaikmodul 1c und gegebenenfalls einem Verbraucher.

Das Monitoring System kann an 4 Punkten einsetzen. Die Spannung des Photovoltaikmoduls sowie der Batterie können jeweils mit einem Spannungssensormodul gemessen werden. Und der Strom der Anlage kann über zwei ACS712 Strommesser abgegriffen werden. Durch eine optionale Sicherung und einem Schalter an der Batterie ist eine Entladung über Nacht ausgeschlossen. Als Problematisch hat sich jedoch die Strommessung an der Batterie erwiesen, wodurch auf eine Betrachtung der Batterie Messwerte zunächst ersteinmal verzichtet wurde.

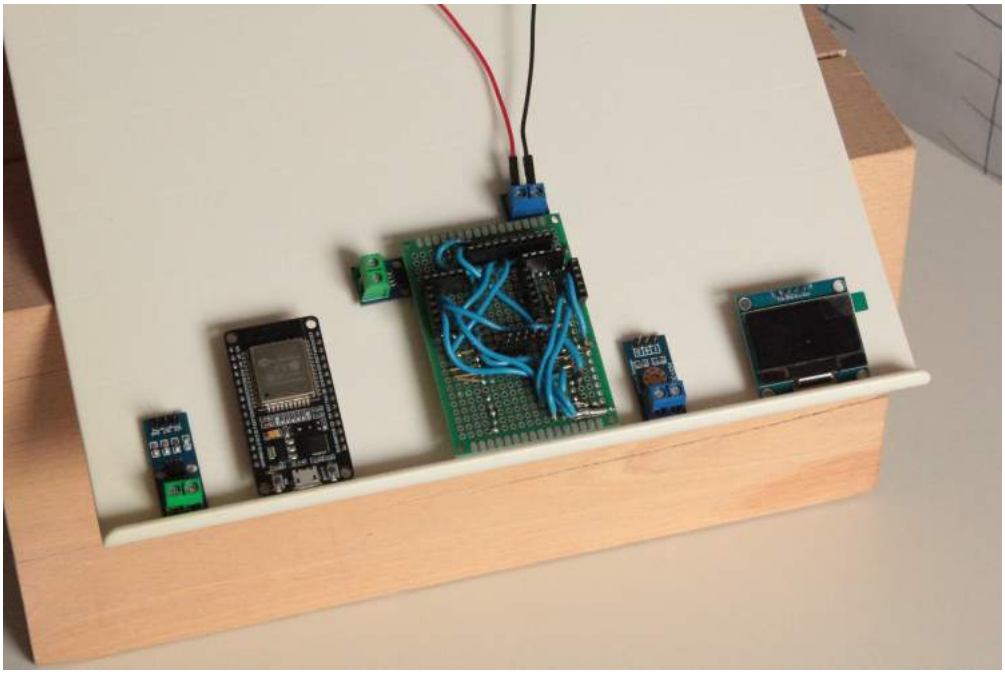


Abbildung 2: Monitoring System in seine Einzelteile zerlegt

Nun wurde ein Monitoring System an die Batterie und die Solaranlage angeschlossen.

Dieses Monitoring System besteht aus 4 Komponenten: Einem ACS712 zur Strommessung, einem Voltagesensor zur Spannungsmessung, einem OLED-Display zur beobachtung vor Ort und dem ESP32 Node MCU als Mikrocontroller.Die Schaltung des Monitoring Systems ist dem folgenden schaubild zu entnehmen 5.
Bevor der Endzustand des Monitoring Systems auf zwei abnehmbare Sensoren festgelegt wurde sollten es vier sein. Zwei für die Batterie und zwei für das Photovoltaikmodul. Doch nach mehreren versuchen die Batteriespannung abzuklemmen wurde die Strom und Spannungsmessung der Batterie aufgegeben. Ein Kurzschluss hat leider eine alte Schaltung zerschossen. Deswegen ist es dabei geblieben nur den Strom und die Spannung des Solarmoduls zu messen. Diese erfolgen wie oben genannt mit einem ACS712 für den Strom und einem Spannungsmessmodul für die Spannung.
Durch den vorfall fällt natürlich die Strom-/Spaannungsmessung an der Batterie weg, sodass die Schaltskizze nun wie im folgenden Bild zu sehen aufgebaut ist4.

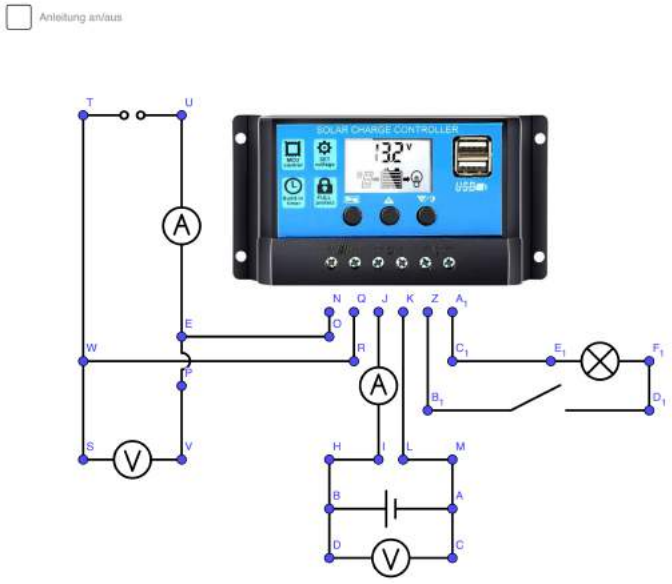


Abbildung 3: Schaltskizze der Solaranlage mit Batteriemessung

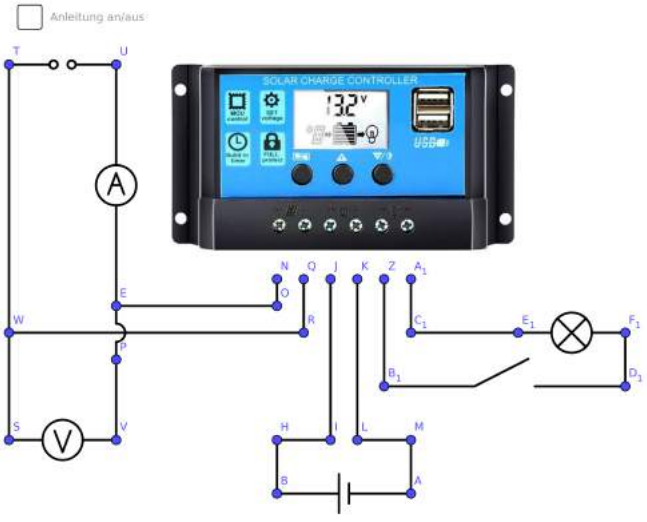


Abbildung 4: Schaltskizze der Solaranlage ohne Batteriemessung

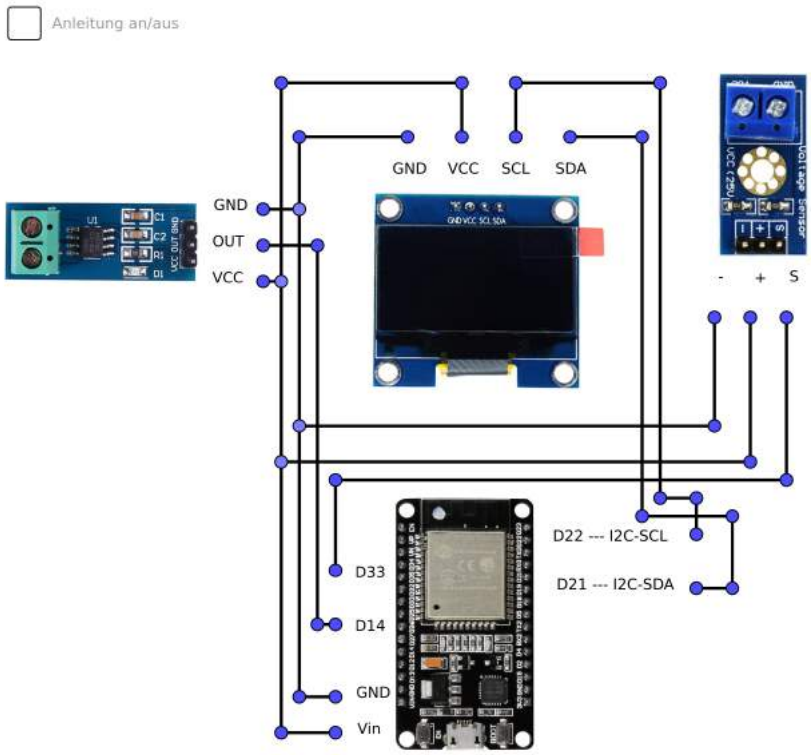


Abbildung 5: Schaltskizze der Monitoring Einheit

2.2 Schaltung

Die Schaltung des Monitoring Systems ist der Abbildung 5 zu entnehmen.

Hier wurde der ESP32 NodeMcu an einen OLED-Display mit 1.3"geschlossen. Die Spannungsversorgung des OLED läuft über das V_{IN} des ESP. Die Sensoren bekommen ihre Betriebsspannung auch aus dieser Quelle. Der SCK (clock Line) Pin(*Arduino - SPI* 2021) ist an den Pin D22 des ESP angeschlossen. Der SDA (data Line) Pin(*Arduino - Wire* 2021) ist an den Pin D21 des ESP angeschlossen. Die Sensoren sind jeweils mit Pin D14 für den Stromsensor und Pin D33 für den Spannungssensor verbunden.

Der wesentliche OLED Code ist im folgenden aufgeführt.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
```

```
#include <Adafruit_SH1106.h>

#define OLED_SDA 21
#define OLED_SCL 22

Adafruit_SH1106 display(21, 22);

void setup()   {
  Serial.begin(115200);
  /* initialize OLED with I2C address 0x3C */
  display.begin(SH1106_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  }

void loop() {
  /* set text size, color, cursor position,
  set buffer with  Hello world and show off*/
  display.setTextSize(2);
  display.setTextColor(WHITE);

display.setCursor(0,0);
  display.print("P= ");
  display.print(solarPower);
  display.print(power);

  display.setCursor(0,20);
  display.print("U= ");
  display.print(solarVoltage);
  display.print(voltage);
  display.print(" ");

  display.setCursor(0,40);
  display.print("I= ");
  display.print(solarCurrent);
  display.print(ampere);

  display.display();
  delay(100);
  display.clearDisplay();
}
```

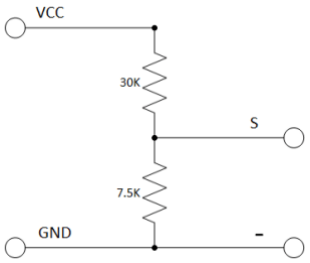


Abbildung 6: Spannungsteiler des Spannungsmessers

2.3 code

Der Programmcode besteht aus zwei Bestandteilen. Zum einen dem Sketch für den Arduino und zum anderen dem HTML-Code für den Webserver.

https://github.com/hdi10/DLN-Solar-Monitoring.git

Arduino Code

Die folgenden Bibliotheken sind bestandteil der Anwendung. Darunter die SPI (für SCK), Wire (für SDA) und die beiden Adafruit Bibliotheken zur Kommunikation mit dem I2C Oled Display.(*Arduino - SPI* 2021; *Arduino - Wire* 2021). Die Spiffs Bibliothek um die index.html Datei des Webservers in HTML und CSS in der Flash Memory der ESP32 zu laden(*Install ESP32 Filesystem Uploader in Arduino IDE | Random Nerd Tutorials* 2018). Und die letzten beiden Bibliotheken WIFi und ESPAsyncWebServer um einen Web Server auf dem port 80 des ESP32 zu generieren.

```
#include <SPI.h>
#include <Wire.h>
#include <SPIFFS.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
```

Daraufhin werden die Sensor Pins deklarier, wobei der Pin D33 [ADC5] des ESP an den Sensor Pin des Spannungsmesser angeschlossen wird und der Pin 14 [ADC16] an den ACS712. Die folgenden Attribute vOut und vIN dienen dem Spannungsmesseralgorithmus zur bestimmung der anliegenden Solarspannung. R1 und R2 sind die Widerstände auf dem Spannungsmesser die in einen Spannungsteiler geschaltet sind. Siehe 6.

```
const int voltageSensor = 33;
const int currentSensor = 14;

float vOUT;
float vIN;
```

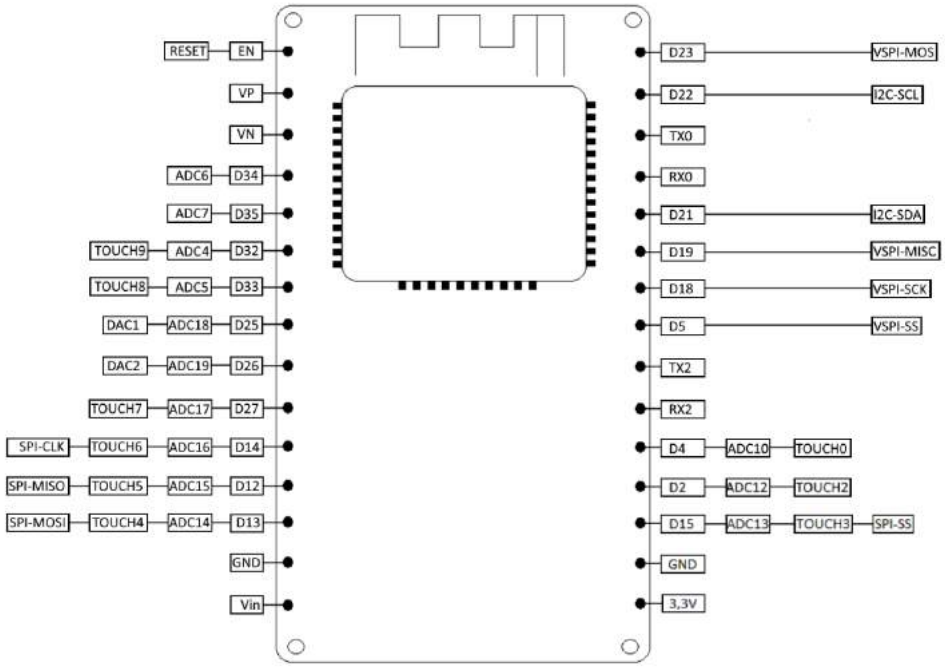


Abbildung 7: Pinout des ESP32 NodeMCU

```
float R1 = 30000.0;
float R2 = 7500.0;
```

Im folgenden sind die drei Methoden aufgeführt um die Spannung, den Strom und die Leistung des Solarmoduls zu berechnen. Bei der Spannunsmessung wird der am Sensor über ein analogRead eingelesene wert über die 12 Bit Auflösung des ADC Pins in 4095 Werte unterteilt, wobei die $3.9V$ die Richtspannung des ESP32 am V_{in} ist. Im folgenden wird über den Spannungsteiler die angelegte Spannung ermittelt. Diese Methode liefert die angelegte Spannung zurück.

```
float spannungsmessung(){
  value = analogRead(voltageSensor);
  vOUT = (value * 3.9) / 4096.0;
  vIN = vOUT / (R2/(R1+R2));
  solarVoltage = vIN;
  delay(50);
  return solarVoltage;
```

```
}
```

Im folgenden wird die Methode zur bestimmung der Stromstärke näher erläutert. Diese bildet zunächts den Mittelwert über 150 Messwerte und unterteilt diesen wieder auf 4096 Werte. Zudem wird eine Abweichung von 2.5 Subtrahiert und durch den Messfehler des 20A ACS712 dividiert. Schlussendlich habe ich eine kleine korrektur vorgenommen um den solarCurrent-Wert in *mA* ausgeben zu können. Diese Methode liefert den Strom der durch das Modul flieSSt zurück.

```
float strommessung() {
unsigned int x=0;
float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;

  for (int x = 0; x < 150; x++){
  AcsValue = analogRead(currentSensor);
  Samples = Samples + AcsValue;
  delay (3);
}
AvgAcs = Samples/150.0;

AcsValueF = (2.5 - (AvgAcs * (3.925 / 4096.0)) )/0.185;
solarCurrent = AcsValueF*(-10);
delay(50);
return solarCurrent;
}
```

Um den Leistungswert des Moduls zu bestimmen bildet diese Methode das Produkt aus der Spannung und der Stromstärke die am Modul anliegt.

```
float leistungsBerechnung(){
  solarPower = solarVoltage*(solarCurrent/1000);
  return solarPower;
}
```

2.4 Drahtlose Übertragung

Die Drahtlose Übertragung wird über zwei Dateien realisiert. Mit dem Arduino Sketch und einer HTML Datei mit css.

Die übertragung der Messwerte der Solaranlage über den ESP32 an den Web Server erfolgt über Wifi. Dazu wird eine HTML Datei in den Flash Speicher des ESP über einen SPIFF upload mihilfe des Arduino ESP32 Filesystem Uploaders(*Install ESP32 Filesystem Uploader in Arduino IDE | Random Nerd Tutorials* 2018).
Mithilfe der Zugangsdaten für das Wifi-Netz wird ein Asynchroner Web Server auf dem Port 80 gestartet.

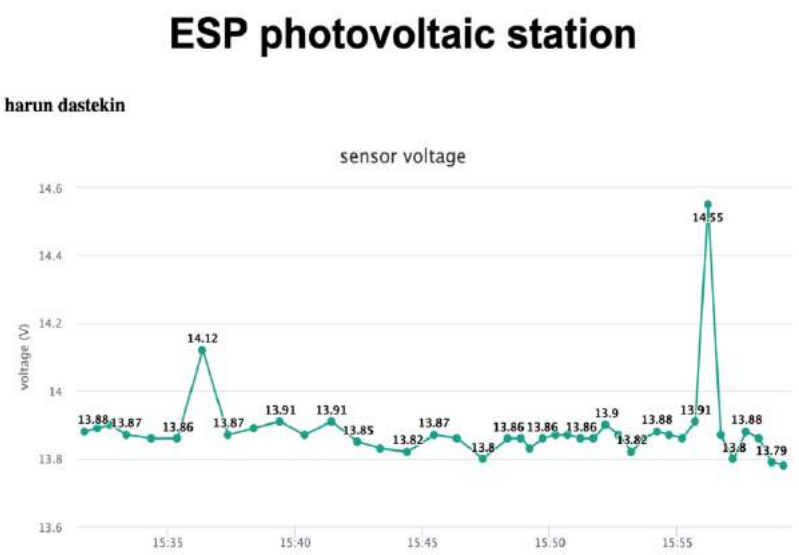


Abbildung 8: Web Server ausschnitt für den Spannungsmesser

```
// Replace with your network credentials
const char* ssid = "Okily_Dokily";
const char* password = "Diablo0704..e-H!88";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
```

Im void setup der Anwendung wird nach dem seriellen Übertragung mit einer Baudrate von 115200, der OLED angesteuert. Da es sich hierbei um einen I2C mit 1.3"mit 128x64 pixel ist wird hier mit 0x3C initialisiert(*OLED Products Category on Adafruit Industries* 2020). Nach dem clearDisplay erfolgt der SPIFFS verbindungsaufbau. Hiernach erfolgt der Wifi verbindungsaufbau und die Ausgabe der IP-Adresse des Netzes.

```
void setup()   {
  Serial.begin(115200);
  display.begin(SH1106_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();

if(!SPIFFS.begin()){
  Serial.println("Fehler bei SPIFF connection");
  while(1);
}

// Connect to Wi-Fi
```

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

// Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
```

Im weiteren verlauf des void setup werden HTTP GET request routen festgelegt und der Server gestartet.

```
// Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html");
  });
  server.on("/voltage", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readVoltage().c_str());
  });
  server.on("/current", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readCurrent().c_str());
  });
  server.on("/power", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", calculatePower().c_str());
  });

  // Start server
  server.begin();
}
```

Zunächst wird die Highcharts Bibliothek importiert (*Interactive JavaScript charts for your webpage | Highcharts* 2021) um ein interaktives Diagramm zu nutzen.

```
<script src="https://code.highcharts.com/highcharts.js"></script>
```

Dann werden für die einzelnen Graphen jeweils eigene Content Division element ( ) mit einer je eigenen id erzeugt.

```
<div id="chart-voltage" class="container"></div>
    <div id="chart-current" class="container"></div>
    <div id="chart-power" class="container"></div>
```

Um Graphen zu generieren und Datenpunkte auf Ihnen zu setzen wird javascript genutzt. Der Folgende Code Snippet legt den Titel, die Achsen, die Farben usw. fest.

```
var chartT = new Highcharts.Chart({
        chart: { renderTo: 'chart-voltage' },
        title: { text: 'sensor voltage' },
        series: [{
            showInLegend: false,
            data: []
        }],
        plotOptions: {
            line: {
                animation: false,
                dataLabels: { enabled: true }
            },
            series: { color: '#059e8a' }
        },
        xAxis: {
            type: 'datetime',
            dateTimeLabelFormats: { second: '%H:%M:%S' }
        },
        yAxis: {
            title: { text: 'voltage (V)' }
        },
        credits: { enabled: false }
```

Schlussendlich fügt die setIntervall Methode dem Graphen Punkte hinzu und schickt alle 30 Sekunden ein GET - request an die /voltage URL um die Spannungswerte des ESP zu bekommen. Die anderen Graphen werden demetsprechend angesprochen.

```
setInterval(function () {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                var x = (new Date()).getTime(),
                    y = parseFloat(this.responseText);

                if (chartT.series[0].data.length > 40) {
                    chartT.series[0].addPoint([x, y], true, true, true);
                } else {
                    chartT.series[0].addPoint([x, y], true, false, true);
                }
            }
        };
        xhttp.open("GET", "/voltage", true);
        xhttp.send();
    }, 30000);
```

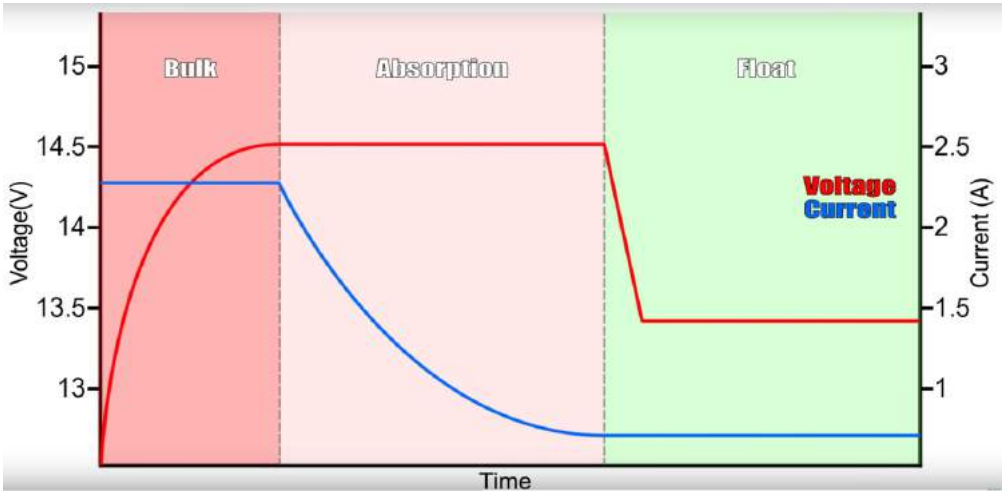


Abbildung 9: Ladekurve der Mppt-Controller Idee

3 Fazit und Ausblick

In Zukunft möchte ich gerne ein Mppt-Regler bauen der drei Zustände besitzt(Electronoobs, 2020). Den "Bulk-Modus"der eine konstante Stromstärke liefert um die Batterie bis auf 80% zu laden (ohne rücksicht auf die Spannung zu nehmen). Ab einer bestimmten Spannung der Batterie soll der nächste Zustand einsetzen Äbsorption". Hier bleibt die Spannung durch den Controller konstant, die Stromstärke nicht mehr, diese fällt mit dem Ladezustand der Batterie (nahe 100%). Dann sobald die Batterie die restlichen 20% geladen hat soll der Controller in den "FloatModus. In diesem Zustand reduziert der Controller die Spannung auf einen festgesetzten Wert und der Stromfluss reduziert sich auf einen Wert kleiner als 1% der Batterie Kapazität. Dadurch kann die Batterie unendlich lange voll geladen bleiben. Ein Bild dieser Idee sieht man auf der folgenden Abbildung9.

Ich habe einen solchen Versuch gestartet. Nach einigen durchgebrannten Schaltungen und fehlenden Teilen habe ich diesen Versuch eingefroren.

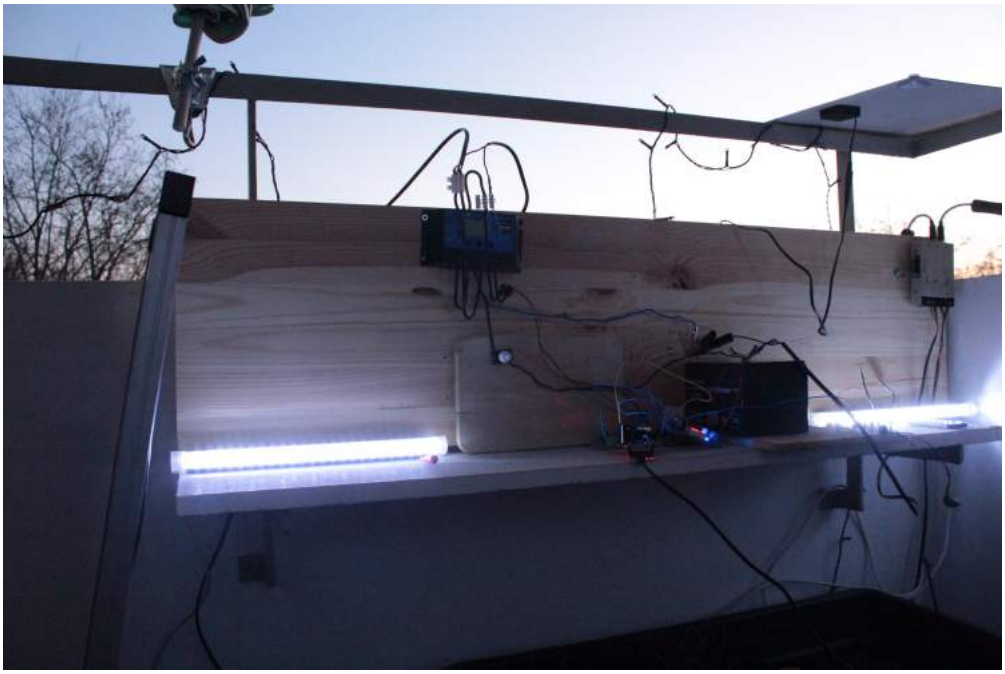


Abbildung 10: Anlage in Konstruktion

Abbildungsverzeichnis

1 Bestandteile der Solaranlage 2
2 Monitoring System in seine Einzelteile zerlegt 3
3 Schaltskizze der Solaranlage mit Batteriemessung 4
4 Schaltskizze der Solaranlage ohne Batteriemessung 4
5 Schaltskizze der Monitoring Einheit 5
6 Spannungsteiler des Spannungsmessers 7
7 Pinout des ESP32 NodeMCU 8
8 Web Server ausschnitt für den Spannungsmesser 10
9 Ladekurve der Mppt-Controller Idee 13
10 Anlage in Konstruktion 14

Literaturverzeichnis

Arduino - SPI. (2021). Verfügbar 26. März 2021 unter https://www.arduino.cc/en/reference/SPI

Arduino - Wire. (2021). Verfügbar 26. März 2021 unter https://www.arduino.cc/en/Reference/Wire

Electronoobs. (2020). MPPT Solar Charger Prototype | 12V Lead-Acid Battery | Bulk Absorption Float. Verfügbar 26. März 2021 unter https://www.youtube.com/watch?v=28WZRZiZuS4&t=824s

Install ESP32 Filesystem Uploader in Arduino IDE | Random Nerd Tutorials. (2018). Verfügbar 26. März 2021 unter https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/

Interactive JavaScript charts for your webpage | Highcharts. (2021). Verfügbar 26. März 2021 unter https://www.highcharts.com/

OLED Products Category on Adafruit Industries. (2020). Verfügbar 26. März 2021 unter https://www.adafruit.com/category/63_98

Solar Panel Data Monitoring using Arduino and LabView. (n. d.). Verfügbar 26. März 2021 unter https://www.hackster.io/Aboubakr_Elhammoumi/solar-panel-data-monitoring-using-arduino-and-labview-8b4c8f

4 Anhang

Arduino Code

```
#include <SPI.h>
#include <Wire.h>
#include <SPIFFS.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>

// Network
const char* ssid = "Okily_Dokily";
const char* password = "Diablo0704..e-H!88";

// AsyncWebServer object on port 80
AsyncWebServer server(80);


#define OLED_SDA 21
#define OLED_SCL 22
#define CURRENT_S 14
#define VOLTAGE_S 33

//////////////////////////////////Attribute//////////////////////////////////
char solar[] = "Solar";
double solarVoltage;
double solarCurrent;

//////////////////////////////////Attribute//////////////////////////////////

const int voltageSensor = 33;
const int currentSensor = 14;

float vOUT;
float vIN;

float solarPower;

float R1 = 30000.0;
float R2 = 7500.0;
int value = 0;


//////////////////////////////////Einheiten//////////////////////////////////
char ampere[] = "mA";
char voltage[] = "V";
char power[] = "W";

////////////////////////////////////////////////////////////////////////////////

Adafruit_SH1106 display(21, 22);

//////////////////////////////////Sensormethoden + Leistung//////////////////////
String readVoltage() {
  return String(spannungsmessung());
}

String readCurrent() {
  return String(strommessung());
}

String calculatePower() {
  return String(leistungsBerechnung());
}
```

```
63
   float leistungsBerechnung(){
65   solarPower = solarVoltage*(solarCurrent/1000);
     return solarPower;
67 }

69 float spannungsmessung(){
     value = analogRead(voltageSensor);
71   vOUT = (value * 3.9) / 4095.0;
     vIN = vOUT / (R2/(R1+R2));
73   solarVoltage = vIN;
     delay(50);
75   return solarVoltage;
   }
77
   float strommessung() {
79 unsigned int x=0;
   float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;
81
     for (int x = 0; x < 150; x++){
83   AcsValue = analogRead(currentSensor);      /
     Samples = Samples + AcsValue;
85   delay (3);
     }
87
     AvgAcs = Samples/150.0;
89   AcsValueF = (2.5 - (AvgAcs * (3.925 / 4096.0)) )/0.185;
     solarCurrent = AcsValueF*(-10);
91
     delay(50);
93
     return solarCurrent;
95 }
   ////////////////////////////////////////////////////////////////////////////////

97
   void setup()   {
99   Serial.begin(115200);
     /* initialize OLED with I2C address 0x3C */
101  display.begin(SH1106_SWITCHCAPVCC, 0x3C);
     display.clearDisplay();
103
   if(!SPIFFS.begin()){
105  Serial.println("Fehler␣bei␣SPIFF␣connection");
     while(1);
107 }

109 // Connect to Wi-Fi
     WiFi.begin(ssid, password);
111  while (WiFi.status() != WL_CONNECTED) {
       delay(1000);
113    Serial.println("Connecting␣to␣WiFi..");
     }
115
   // Print ESP32 Local IP Address
117  Serial.println(WiFi.localIP());

119 // Route for root / web page
     server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
121    request->send(SPIFFS, "/index.html");
     });
123  server.on("/voltage", HTTP_GET, [](AsyncWebServerRequest *request){
       request->send_P(200, "text/plain", readVoltage().c_str());
125  });
     server.on("/current", HTTP_GET, [](AsyncWebServerRequest *request){
127    request->send_P(200, "text/plain", readCurrent().c_str());
     });
129  server.on("/power", HTTP_GET, [](AsyncWebServerRequest *request){
       request->send_P(200, "text/plain", calculatePower().c_str());
```

```
131     });
133     // Start server
        server.begin();
135 }
137 void loop() {
        display.setTextSize(2);
139     display.setTextColor(WHITE);
141     solarCurrent = strommessung();
        solarVoltage = spannungsmessung();
143     solarPower = leistungsBerechnung();
145
        display.setCursor(0,0);
147     display.print("P=␣");
        display.print(solarPower);
149     display.print(power);
151     display.setCursor(0,20);
        display.print("U=␣");
153     display.print(solarVoltage);
        display.print(voltage);
155     display.print("␣");
157     display.setCursor(0,40);
        display.print("I=␣");
159     display.print(solarCurrent);
        display.print(ampere);
161
        display.display();
163     delay(100);
        display.clearDisplay();
165 }
```

Webserver Code

```
1  <!DOCTYPE HTML>
   <html>
3
   <head>
5      <meta name="viewport" content="width=device-width,␣initial-scale=1">
       <script src="https://code.highcharts.com/highcharts.js"></script>
7      <style>
           body {
9              min-width: 310px;
               max-width: 800px;
11             height: 400px;
               margin: 0 auto;
13         }
15         h2 {
               font-family: Arial;
17             font-size: 2.5rem;
               text-align: center;
19         }
       </style>
21 </head>
23 <body>
       <h2>ESP photovoltaic station</h2>
25     <h3>harun dastekin</h3>
       <div id="chart-voltage" class="container"></div>
27     <div id="chart-current" class="container"></div>
       <div id="chart-power" class="container"></div>
```

```
</body>
<script>
    var chartT = new Highcharts.Chart({
        chart: { renderTo: 'chart-voltage' },
        title: { text: 'sensor voltage' },
        series: [{
            showInLegend: false,
            data: []
        }],
        plotOptions: {
            line: {
                animation: false,
                dataLabels: { enabled: true }
            },
            series: { color: '#059e8a' }
        },
        xAxis: {
            type: 'datetime',
            dateTimeLabelFormats: { second: '%H:%M:%S' }
        },
        yAxis: {
            title: { text: 'voltage (V)' }

        },
        credits: { enabled: false }
    });
    setInterval(function () {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                var x = (new Date()).getTime(),
                    y = parseFloat(this.responseText);
                //console.log(this.responseText);
                if (chartT.series[0].data.length > 40) {
                    chartT.series[0].addPoint([x, y], true, true, true);
                } else {
                    chartT.series[0].addPoint([x, y], true, false, true);
                }
            }
        };
        xhttp.open("GET", "/voltage", true);
        xhttp.send();
    }, 30000);

    var chartH = new Highcharts.Chart({
        chart: { renderTo: 'chart-current' },
        title: { text: 'ACS712 current' },
        series: [{
            showInLegend: false,
            data: []
        }],
        plotOptions: {
            line: {
                animation: false,
                dataLabels: { enabled: true }
            }
        },
        xAxis: {
            type: 'datetime',
            dateTimeLabelFormats: { second: '%H:%M:%S' }
        },
        yAxis: {
            title: { text: 'current (mA)' }
        },
        credits: { enabled: false }
    });
    setInterval(function () {
        var xhttp = new XMLHttpRequest();
```

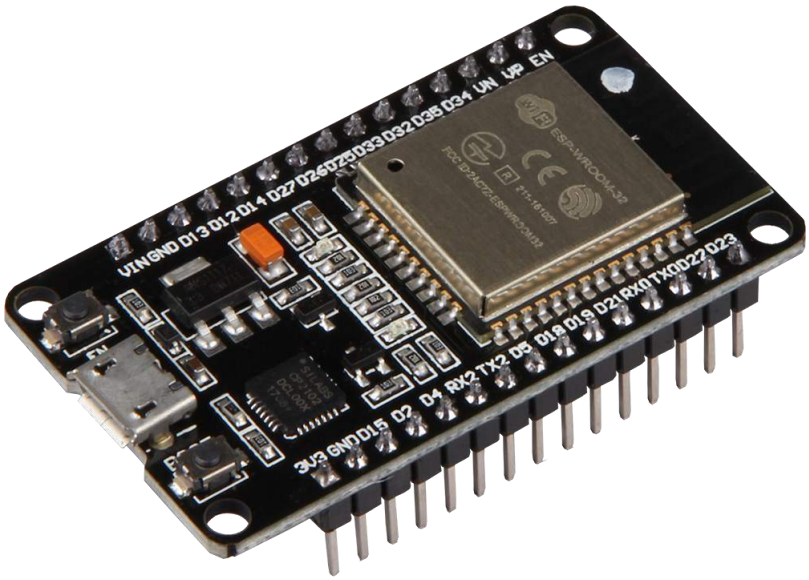
```
97          xhttp.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
99                  var x = (new Date()).getTime(),
                        y = parseFloat(this.responseText);
101                 //console.log(this.responseText);
                    if (chartH.series[0].data.length > 40) {
103                     chartH.series[0].addPoint([x, y], true, true, true);
                    } else {
105                     chartH.series[0].addPoint([x, y], true, false, true);
                    }
107             }
            };
109         xhttp.open("GET", "/current", true);
            xhttp.send();
111     }, 30000);

113     var chartP = new Highcharts.Chart({
            chart: { renderTo: 'chart-power' },
115         title: { text: 'Solar power' },
            series: [{
117             showInLegend: false,
                data: []
119         }],
            plotOptions: {
121             line: {
                    animation: false,
123                 dataLabels: { enabled: true }
                },
125             series: { color: '#18009c' }
            },
127         xAxis: {
                type: 'datetime',
129             dateTimeLabelFormats: { second: '%H:%M:%S' }
            },
131         yAxis: {
                title: { text: 'power (W)' }
133         },
            credits: { enabled: false }
135     });
        setInterval(function () {
137         var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function () {
139             if (this.readyState == 4 && this.status == 200) {
                    var x = (new Date()).getTime(),
141                     y = parseFloat(this.responseText);
                    //console.log(this.responseText);
143                 if (chartP.series[0].data.length > 40) {
                        chartP.series[0].addPoint([x, y], true, true, true);
145                 } else {
                        chartP.series[0].addPoint([x, y], true, false, true);
147                 }
                }
149         };
            xhttp.open("GET", "/power", true);
151         xhttp.send();
        }, 30000);
153 </script>

155 </html>
```

NODEMCU ESP32

Microcontroller Entwicklungsplatine



1. ALLGEMEINE INFORMATIONEN

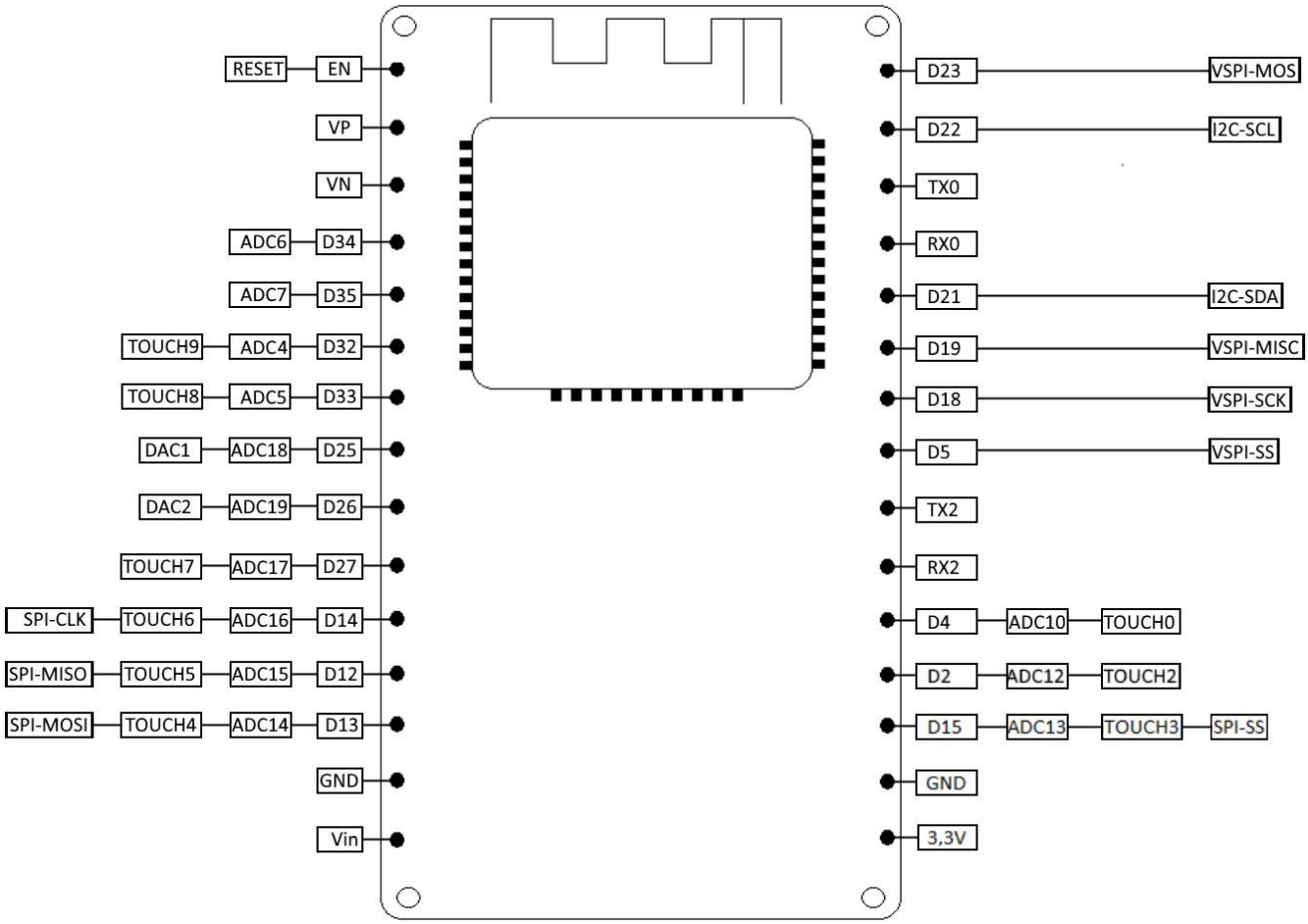
Sehr geehrter Kunde,

vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. ÜBERSICHT

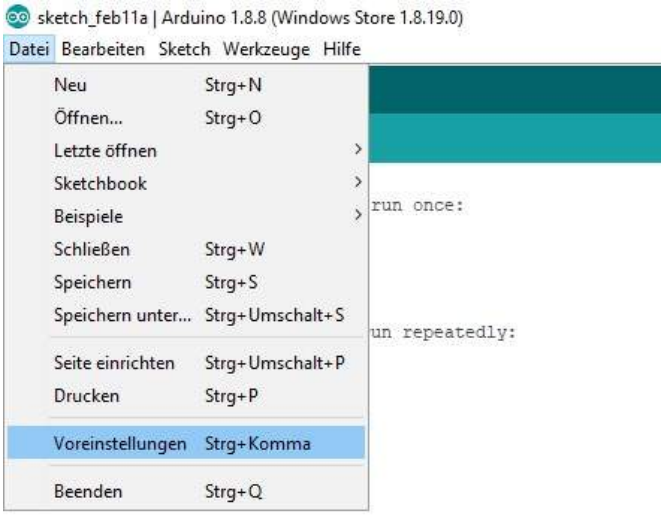
Das NodeMCU ESP32 Modul ist ein kompaktes Prototyping-Board und lässt sich bequem über die Arduino IDE programmieren. Es verfügt über 2,4 GHz Dual-Mode WiFi und Bluetooth. Ebenfalls auf der Microcontroller Entwicklungsplatine integriert sind: 512 kB SRAM und 4 MB Speicher, 2x DAC, 15x ADC, 1x SPI, 1x I^2C, 2x UART. PWM ist an jedem digitalen Pin aktiviert.
Eine Übersicht über die vorhandenen Pins können Sie der folgenden Abbildung entnehmen:



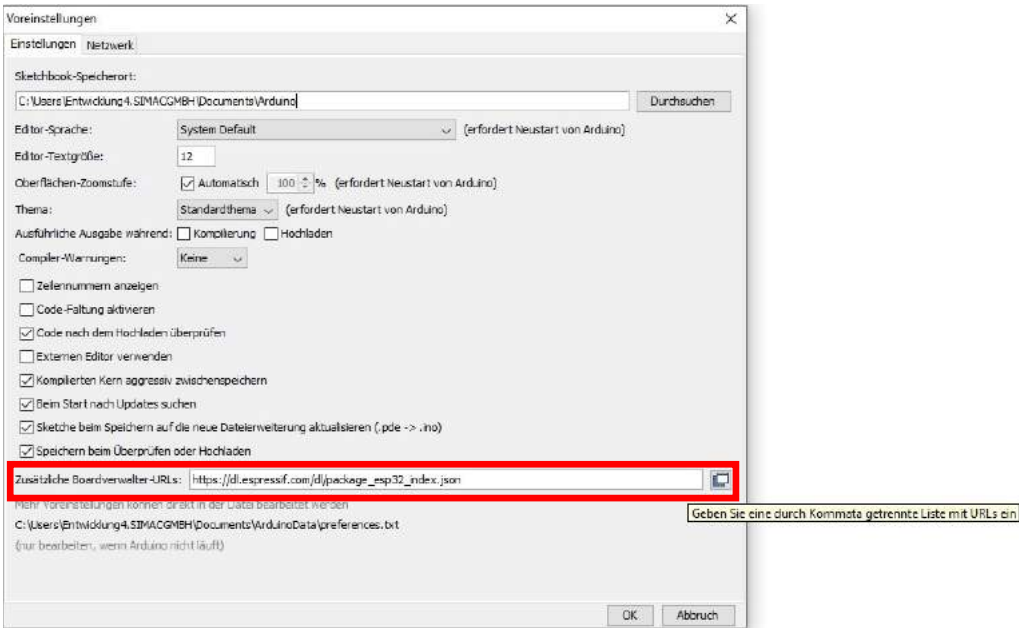
3. INSTALLATION DER MODULE

Falls Sie die Arduino IDE noch nicht auf Ihrem Computer installiert haben, laden Sie diese zunächst herunter und installieren Sie diese.
Laden Sie sich nun die aktualisierten CP210x USB-UART Treiber für Ihr Betriebssystem herunter und installieren Sie diese. Als nächstes müssen Sie einen neuen Boardverwalter hinzufügen, befolgen Sie dafür die folgenden Schritte.

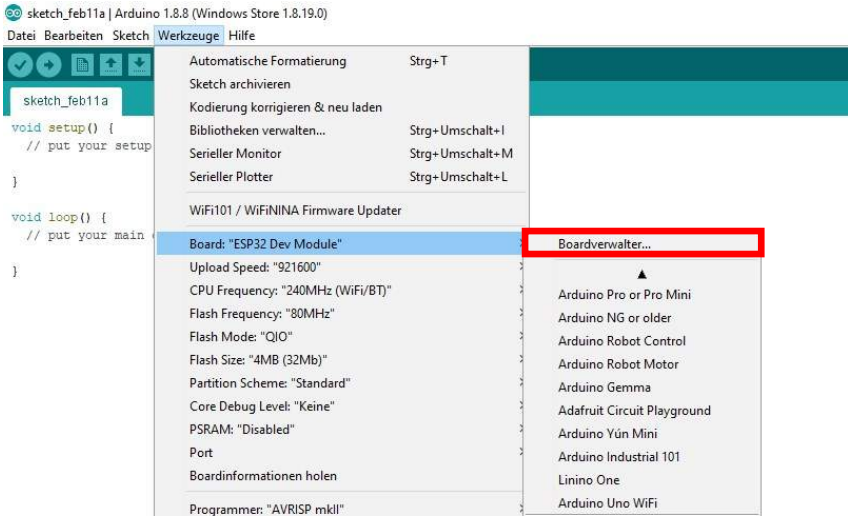
1. Klicken Sie auf Datei → Voreinstellungen



2. Fügen Sie bei zusätzliche Boardverwalter-URLs den folgenden Link ein:
   https://dl.espressif.com/dl/package_esp32_index.json
   Mehrere URLs können Sie durch ein Komma trennen.

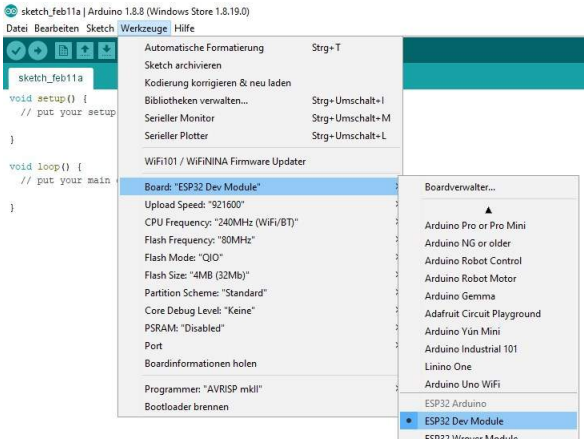


3. Klicken Sie auf Werkzeuge → Board → Boardverwalter…



4. Installieren Sie ***esp32 by Espressif Systems***.

Die Installation ist nun abgeschlossen. Sie können jetzt unter Werkzeuge → Board das **ESP32 Dev Module** auswählen.



Achtung! Nach der Erstinstallation kann sich die Baudrate unter Umständen auf *921600* geändert haben. Dies führt möglicherweise zu Problemen. Wählen Sie in diesem Fall die Baudrate *115200* um eventuelle Probleme zu vermeiden.

4. VERWENDUNG

Ihr NodeMCU ESP32 ist nun bereit zur Verwendung. Schließen Sie es einfach mit einem USB-Kabel an Ihren Computer an.
Die installierte Bibliothek stellt bereits viele Beispiele zur Verfügung um einen schnellen Einblick in das Modul zu ermöglichen.
Die Beispiele finden Sie in Ihrer Arduino IDE unter Datei → Beispiele→ ESP32.

Der schnellste und einfachste Weg um Ihren NodeMCU ESP32 zu testen, ist der Abruf der Gerätenummer. Kopieren Sie entweder den nachfolgenden Code oder verwenden Sie das Beispiel **GetChipID** aus der Arduino IDE:

```
uint64_t chipid;

void setup() {
     Serial.begin(115200);
}

void loop() {
     chipid=ESP.getEfuseMac();//The chip ID is essentially its MAC address(length:6bytes)
     Serial.printf("ESP32 Chip ID = %04X",(uint16_t)(chipid>>32));//print High 2 bytes
     Serial.printf("%08X\n",(uint32_t)chipid);//print Low 4bytes.

     delay(3000);

}
```

Zum Hochladen müssen Sie auf die Schaltfläche **Hochladen** von der Arduino IDE klicken und die Taste **BOOT** auf dem SBC-NodeMCU-ESP32 gedrückt halten. Das Hochladen ist abgeschlossen, bis das Beschreiben 100% erreicht hat und Sie aufgefordert werden einen Neustart durchzuführen (Hard-Resetting via RTS Pin...) mit der Taste **EN**.
Die Ausgabe des Beispielprogramms können Sie im seriellen Monitor aufrufen.

5. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in Haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
Simac GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

6. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: 20.03.2020

ACS712

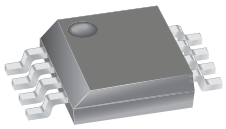
Fully Integrated, Hall-Effect-Based Linear Current Sensor IC with 2.1 kVRMS Isolation and a Low-Resistance Current Conductor

FEATURES AND BENEFITS

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 µs output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at T_A = 25°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kVRMS minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

TÜV America
Certificate Number:
U8V 15 05 54214 038
CB 13 06 54214 026

Package: 8-Lead SOIC (suffix LC)



Not to scale

DESCRIPTION

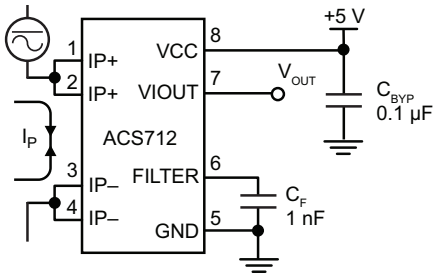
The Allegro™ ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switch-mode power supplies, and overcurrent fault protection. The device is not intended for automotive applications.

The device consists of a precise, low-offset, linear Hall circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which the Hall IC converts into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sampling. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power loss. The thickness of the copper conductor allows survival of

Continued on the next page…

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT}. that varies linearly with the uni- or bi-directional AC or DC primary sampled current, I_P, within the range specified. C_F is recommended for noise management, with values that depend on the application.

DESCRIPTION (continued)

the device at up to 5× overcurrent conditions. The terminals of the conductive path are electrically isolated from the signal leads (pins 5 through 8). This allows the ACS712 to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 package. The leadframe is plated with 100% matte tin, which is compatible with standard lead (Pb) free printed circuit board assembly processes. Internally, the device is Pb-free, except for flip-chip high-temperature Pb-based solder balls, currently exempt from RoHS. The device is fully calibrated prior to shipment from the factory.

SELECTION GUIDE

Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

*Contact Allegro for additional packing options.

ABSOLUTE MAXIMUM RATINGS

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
Maximum Junction Temperature	T_J(max)		165	°C
Storage Temperature	T_{stg}		–65 to 170	°C

ISOLATION CHARACTERISTICS

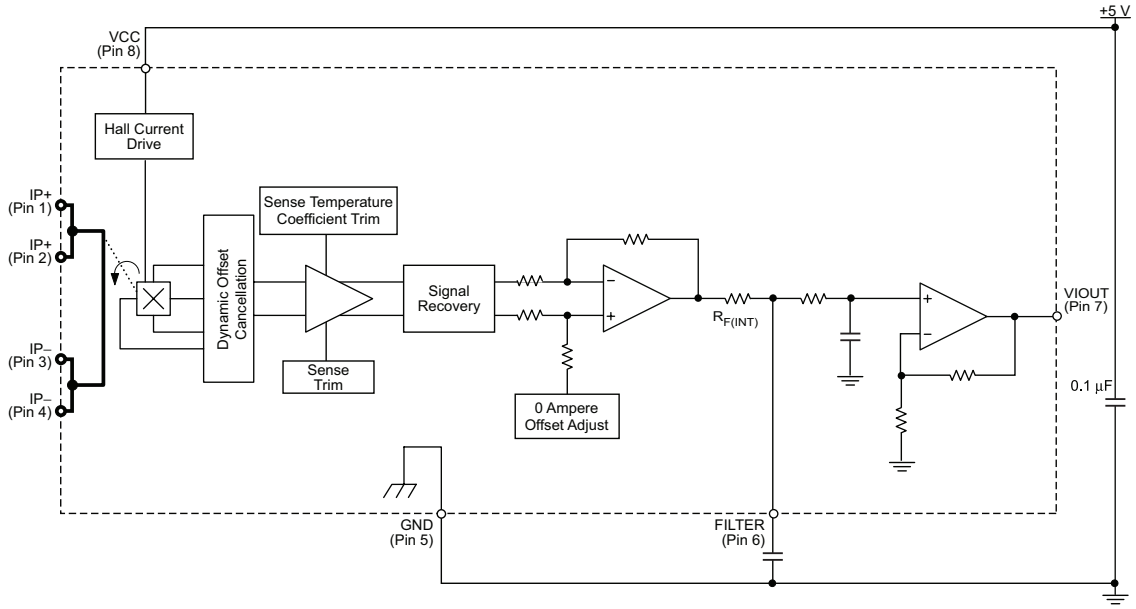
Characteristic	Symbol	Notes	Rating	Unit
Dielectric Strength Test Voltage*	V_{ISO}	Agency type-tested for 60 seconds per UL standard 60950-1, 1st Edition	2100	VAC
Working Voltage for Basic Isolation	V_{WFSI}	For basic (single) isolation per UL standard 60950-1, 1st Edition	354	VDC or V_{pk}
Working Voltage for Reinforced Isolation	V_{WFRI}	For reinforced (double) isolation per UL standard 60950-1, 1st Edition	184	VDC or V_{pk}

* Allegro does not conduct 60-second testing. It is done only during the UL certification process.

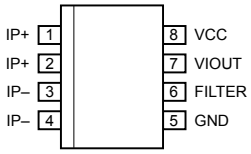
Parameter	Specification
Fire and Electric Shock	CAN/CSA-C22.2 No. 60950-1-03 UL 60950-1:2003 EN 60950-1:2001

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

FUNCTIONAL BLOCK DIAGRAM



Pinout Diagram



Terminal List

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

COMMON OPERATING CHARACTERISTICS [1]: Over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	$I_P = I_P$(max), T_A = 25°C, C_{OUT} = open	–	3.5	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	µs
Magnetic Coupling [2]			–	12	–	G/A
Internal Filter Resistance [3]	$R_{F(INT)}$			1.7		kΩ

[1] Device may be operated at higher primary current levels, I_P, and ambient, T_A, and internal leadframe temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2] 1G = 0.1 mT.

[3] $R_{F(INT)}$ forms an RC circuit via the FILTER pin.

COMMON THERMAL CHARACTERISTICS [1]

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Operating Internal Leadframe Temperature	T_A	E range	–40	–	85	°C
Characteristic	**Symbol**	**Test Conditions**			**Value**	**Units**
Junction-to-Lead Thermal Resistance [2]	$R_{\theta JL}$	Mounted on the Allegro ASEK 712 evaluation board			5	°C/W
Junction-to-Ambient Thermal Resistance	$R_{\theta JA}$	Mounted on the Allegro 85-0322 evaluation board, includes the power consumed by the board			23	°C/W

[1] Additional thermal information is available on the Allegro website.

[2] The Allegro evaluation board has 1500 mm^2 of 2 oz. copper on each side, connected to pins 1 and 2, and to pins 3 and 4, with thermal vias connecting the layers. Performance values include the power consumed by the PCB. Further details on the board are available from the Frequently Asked Questions document on our website. Further information about board design and thermal performance also can be found in the Applications Information section of this datasheet.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

x05B PERFORMANCE CHARACTERISTICS [1] : T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–5	–	5	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	180	185	190	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 185 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	21	–	mV
Zero Current Output Slope	$\Delta V_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.26	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.054	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.008	–	mV/A/°C
Total Output Error [2]	E_{TOT}	I_P =±5 A, T_A = 25°C	–	±1.5	–	%

[1] Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, $T_{J(max)}$, is not exceeded.
[2] Percentage of I_P, with I_P = 5 A. Output filtered.

x20A PERFORMANCE CHARACTERISTICS [1] T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–20	–	20	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	96	100	104	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 100 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	11	–	mV
Zero Current Output Slope	$\Delta V_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.34	–	mV/°C
		T_A = 25°C to 150°C	–	–0.07	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.017	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.004	–	mV/A/°C
Total Output Error [2]	E_{TOT}	I_P =±20 A, T_A = 25°C	–	±1.5	–	%

[1] Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.
[2] Percentage of I_P, with I_P = 20 A. Output filtered.

x30A PERFORMANCE CHARACTERISTICS [1]: T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–30	–	30	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	63	66	69	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 66 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	7	–	mV
Zero Current Output Slope	$\Delta V_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.35	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.007	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.002	–	mV/A/°C
Total Output Error [2]	E_{TOT}	I_P = ±30 A, T_A = 25°C	–	±1.5	–	%

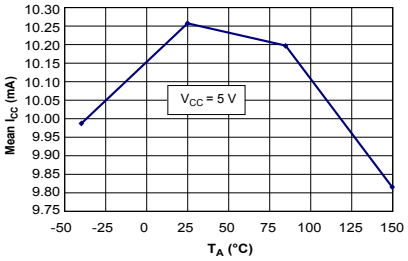
[1] Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.
[2] Percentage of I_P, with I_P = 30 A. Output filtered.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

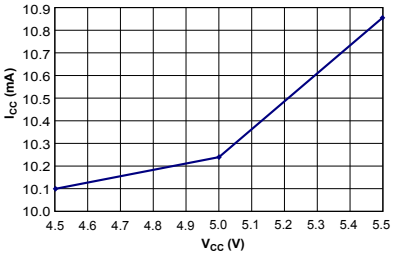
CHARACTERISTIC PERFORMANCE

I_P = 5 A, unless otherwise specified

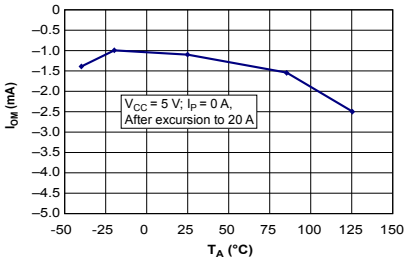
Mean Supply Current versus Ambient Temperature



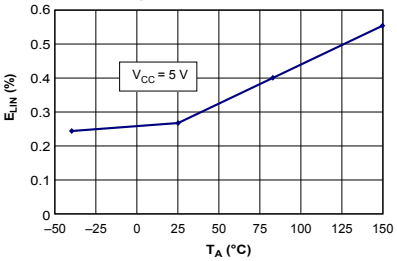
Supply Current versus Supply Voltage



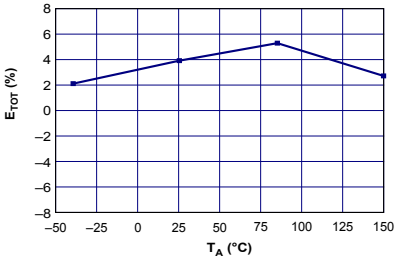
Magnetic Offset versus Ambient Temperature



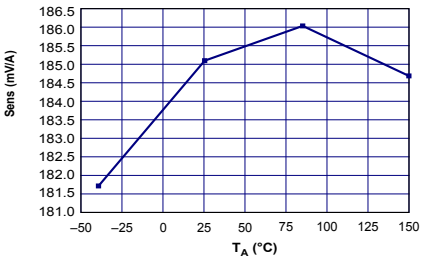
Nonlinearity versus Ambient Temperature



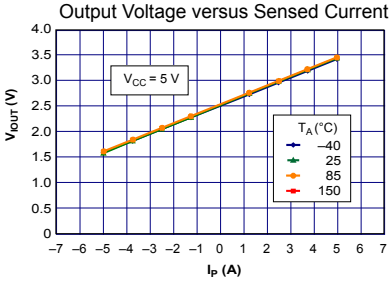
Mean Total Output Error versus Ambient Temperature



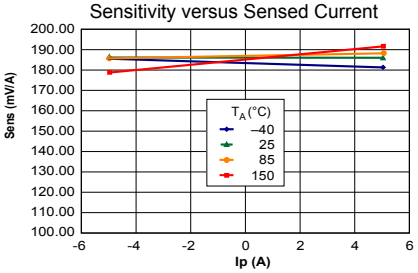
Sensitivity versus Ambient Temperature



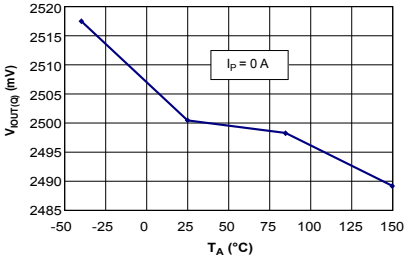
Output Voltage versus Sensed Current



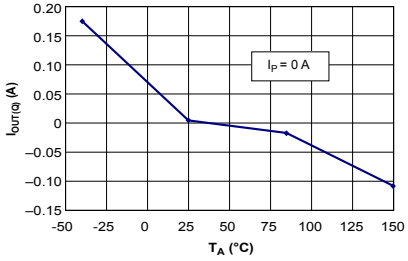
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature

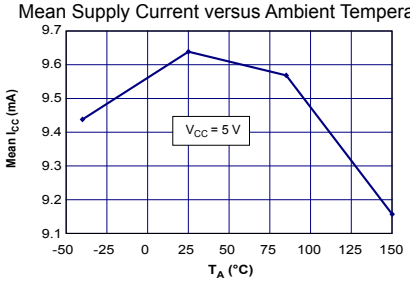


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

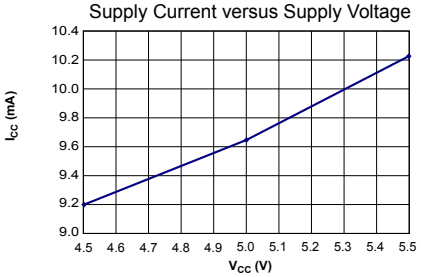
CHARACTERISTIC PERFORMANCE

I_P = 20 A, unless otherwise specified

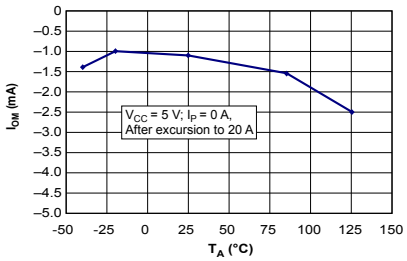
Mean Supply Current versus Ambient Temperature



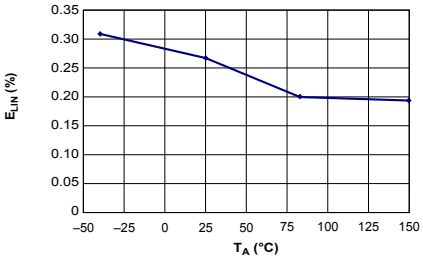
Supply Current versus Supply Voltage



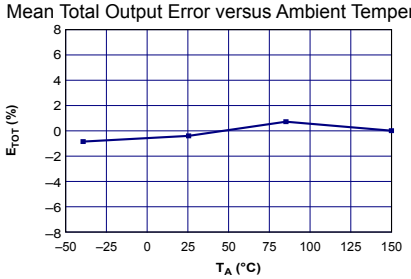
Magnetic Offset versus Ambient Temperature



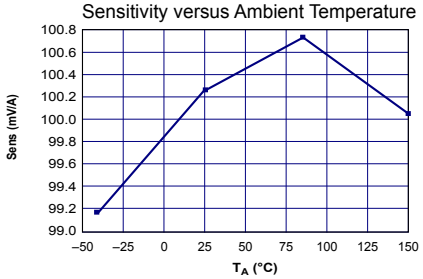
Nonlinearity versus Ambient Temperature



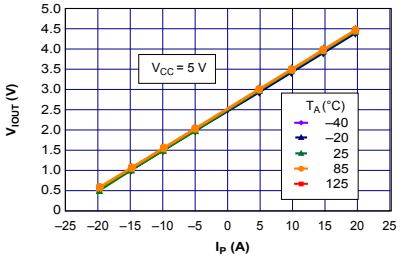
Mean Total Output Error versus Ambient Temperature



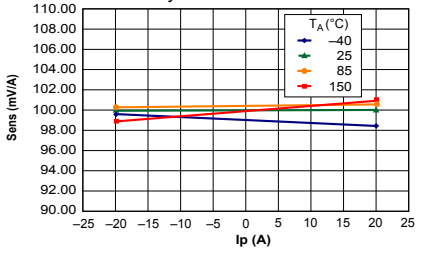
Sensitivity versus Ambient Temperature



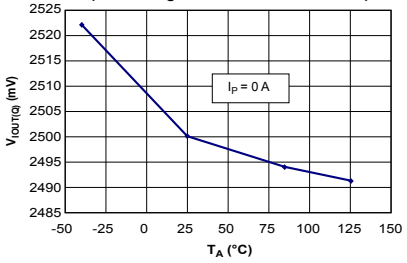
Output Voltage versus Sensed Current



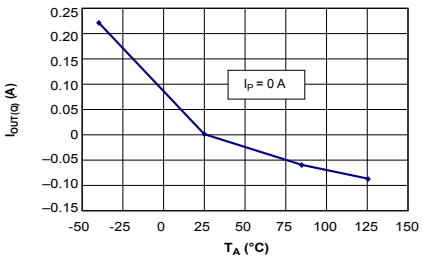
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature

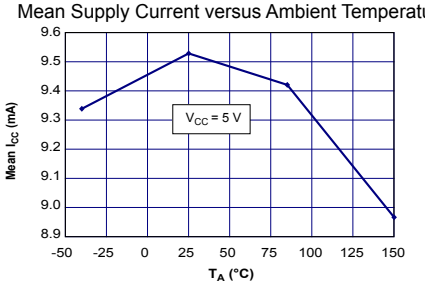


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

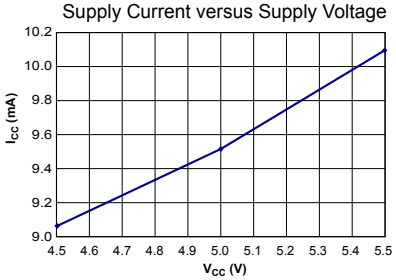
CHARACTERISTIC PERFORMANCE

I_P = 30 A, unless otherwise specified

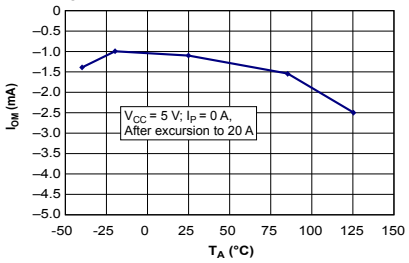
Mean Supply Current versus Ambient Temperature



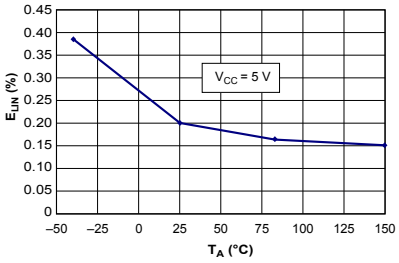
Supply Current versus Supply Voltage



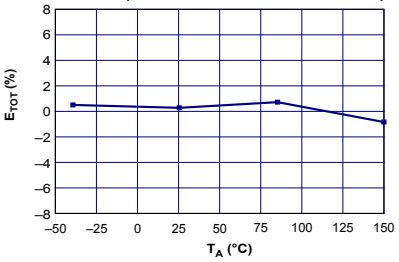
Magnetic Offset versus Ambient Temperature



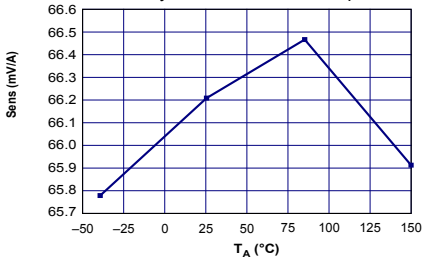
Nonlinearity versus Ambient Temperature



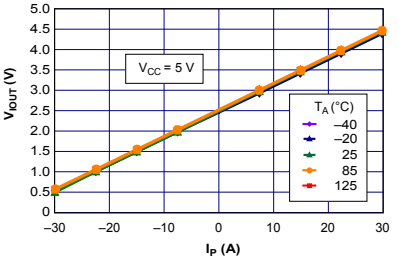
Mean Total Output Error versus Ambient Temperature



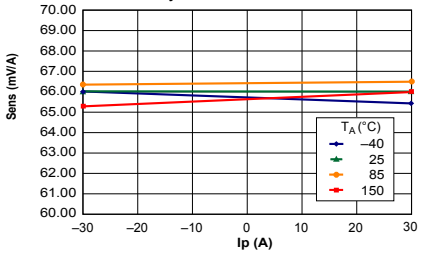
Sensitivity versus Ambient Temperature



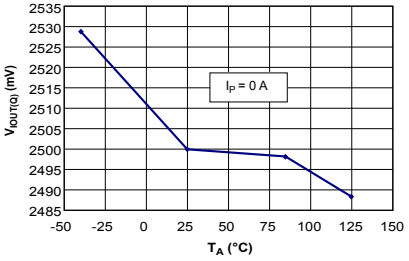
Output Voltage versus Sensed Current



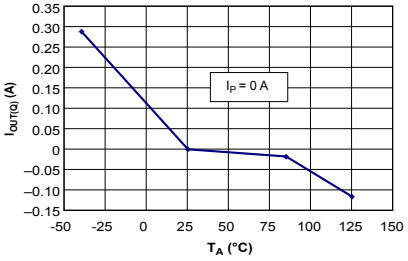
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature



Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

DEFINITIONS OF ACCURACY CHARACTERISTICS

Sensitivity (Sens). The change in device output in response to a 1 A change through the primary conductor. The sensitivity is the product of the magnetic circuit sensitivity (G/A) and the linear IC amplifier gain (mV/G). The linear IC amplifier gain is programmed at the factory to optimize the sensitivity (mV/A) for the full-scale current of the device.

Noise (V_{NOISE}). The product of the linear IC amplifier gain (mV/G) and the noise floor for the Allegro Hall effect linear IC (≈1 G). The noise floor is derived from the thermal and shot noise observed in Hall elements. Dividing the noise (mV) by the sensitivity (mV/A) provides the smallest current that the device is able to resolve.

Linearity (E_{LIN}). The degree to which the voltage output from the IC varies in direct proportion to the primary current through its full-scale amplitude. Nonlinearity in the output can be attributed to the saturation of the flux concentrator approaching the full-scale current. The following equation is used to derive the linearity:

$$100\left\{1-\left[\frac{\Delta\text{ gain}\times\%\text{ sat}\,(V_{\text{IOUT\_full-scale amperes}}-V_{\text{IOUT(Q)}})}{2\,(V_{\text{IOUT\_half-scale amperes}}-V_{\text{IOUT(Q)}})}\right]\right\}$$

where $V_{\text{IOUT\_full-scale amperes}}$ = the output voltage (V) when the sampled current approximates full-scale $\pm I_P$.

Symmetry (E_{SYM}). The degree to which the absolute voltage output from the IC varies in proportion to either a positive or negative full-scale primary current. The following formula is used to derive symmetry:

$$100\left(\frac{V_{\text{IOUT\_+ full-scale amperes}}-V_{\text{IOUT(Q)}}}{V_{\text{IOUT(Q)}}-V_{\text{IOUT\_–full-scale amperes}}}\right)$$

Quiescent output voltage ($V_{IOUT(Q)}$). The output of the device when the primary current is zero. For a unipolar supply voltage, it nominally remains at $V_{CC}/2$. Thus, V_{CC} = 5 V translates into $V_{IOUT(Q)}$ = 2.5 V. Variation in $V_{IOUT(Q)}$ can be attributed to the resolution of the Allegro linear IC quiescent voltage trim and thermal drift.

Electrical offset voltage (V_{OE}). The deviation of the device output from its ideal quiescent value of $V_{CC}/2$ due to nonmagnetic causes. To convert this voltage to amperes, divide by the device sensitivity, Sens.

Accuracy (E_{TOT}). The accuracy represents the maximum deviation of the actual output from its ideal value. This is also known as the total output error. The accuracy is illustrated graphically in the output voltage versus current chart at right.

Accuracy is divided into four areas:

- **0 A at 25°C.** Accuracy at the zero current flow at 25°C, without the effects of temperature.
- **0 A over Δ temperature.** Accuracy at the zero current flow including temperature effects.
- **Full-scale current at 25°C.** Accuracy at the the full-scale current at 25°C, without the effects of temperature.
- **Full-scale current over Δ temperature.** Accuracy at the full-scale current flow including temperature effects.

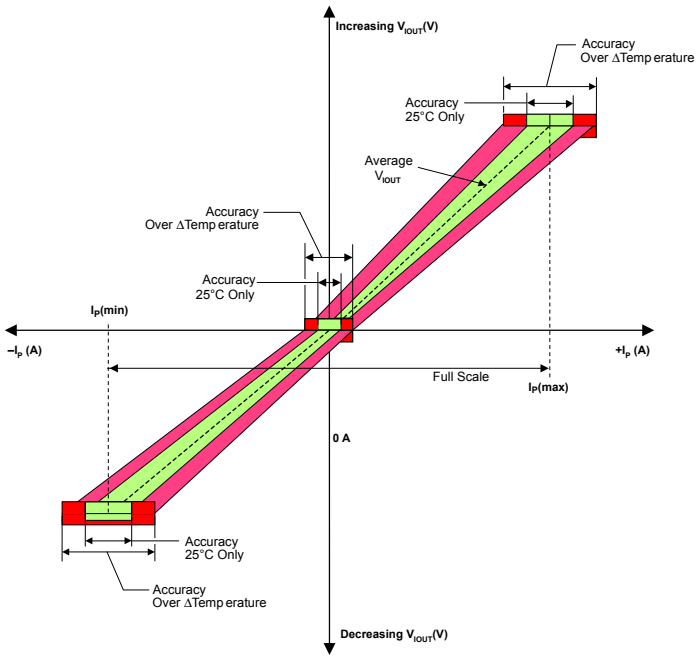
Ratiometry. The ratiometric feature means that its 0 A output, $V_{IOUT(Q)}$, (nominally equal to $V_{CC}/2$) and sensitivity, Sens, are proportional to its supply voltage, V_{CC}. The following formula is used to derive the ratiometric change in 0 A output voltage, $\Delta V_{IOUT(Q)RAT}$ (%).

$$100\left(\frac{V_{\text{IOUT(Q)VCC}}/V_{\text{IOUT(Q)5V}}}{V_{CC}/5\text{ V}}\right)$$

The ratiometric change in sensitivity, $\Delta Sens_{RAT}$ (%), is defined as:

$$100\left(\frac{Sens_{VCC}/Sens_{5V}}{V_{CC}/5\text{ V}}\right)$$

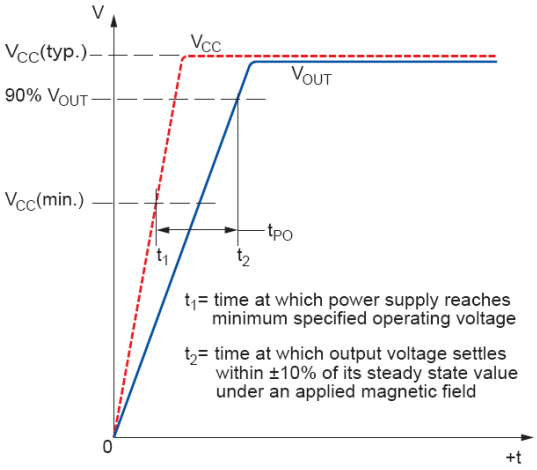
Output Voltage versus Sampled Current
Accuracy at 0 A and at Full-Scale Current



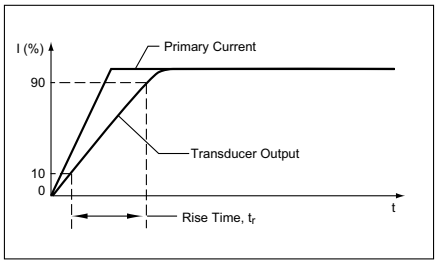
Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

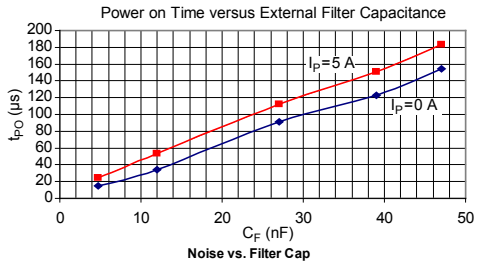
DEFINITIONS OF DYNAMIC RESPONSE CHARACTERISTICS

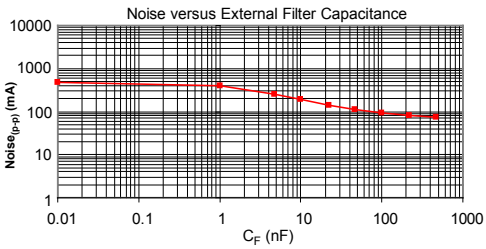
Power-On Time (t_{PO}). When the supply is ramped to its operating voltage, the device requires a finite time to power its internal components before responding to an input magnetic field. Power-On Time, t_{PO} , is defined as the time it takes for the output voltage to settle within ±10% of its steady state value under an applied magnetic field, after the power supply has reached its minimum specified operating voltage, V_{CC}(min), as shown in the chart at right.

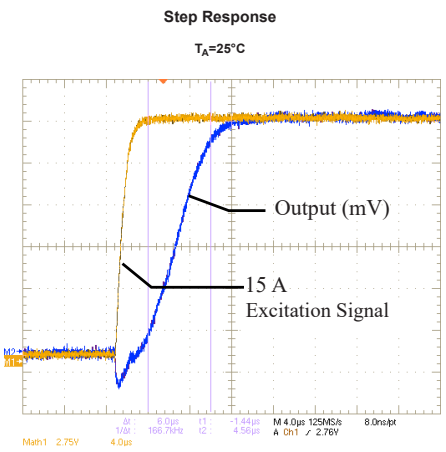


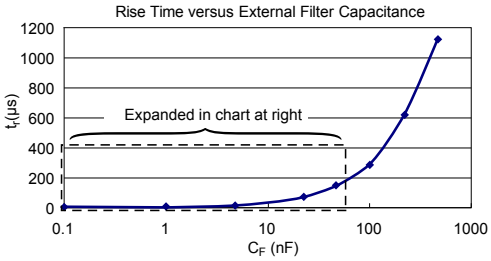
Rise time (t_r). The time interval between a) when the device reaches 10% of its full scale value, and b) when it reaches 90% of its full scale value. The rise time to a step response is used to derive the bandwidth of the device, in which $f(-3\ dB) = 0.35/t_r$. Both t_r and $t_{RESPONSE}$ are detrimentally affected by eddy current losses observed in the conductive IC ground plane.



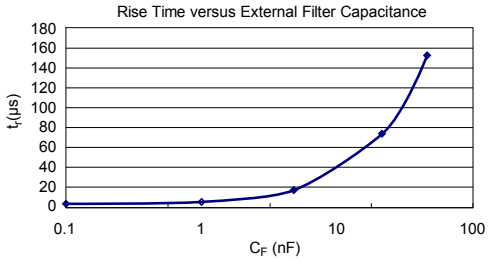








C_F (nF)	t_r (µs)
Open	3.5
1	5.8
4.7	17.5
22	73.5
47	88.2
100	291.3
220	623
470	1120

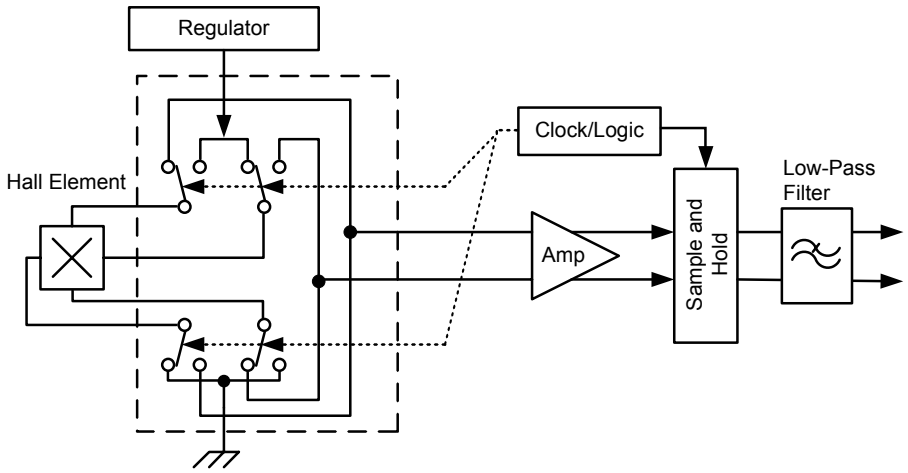


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

CHOPPER STABILIZATION TECHNIQUE

Chopper Stabilization is an innovative circuit technique that is used to minimize the offset voltage of a Hall element and an associated on-chip amplifier. Allegro has a Chopper Stabilization technique that nearly eliminates Hall IC output drift induced by temperature or package stress effects. This offset reduction technique is based on a signal modulation-demodulation process. Modulation is used to separate the undesired DC offset signal from the magnetically induced signal in the frequency domain. Then, using a low-pass filter, the modulated DC offset is suppressed while the magnetically induced signal passes through the filter. As a result of this chopper stabilization approach, the output voltage from the Hall IC is desensitized to the effects of temperature and mechanical stress. This technique produces devices that have an extremely stable Electrical Offset Voltage, are immune to thermal stress, and have precise recoverability after temperature cycling.

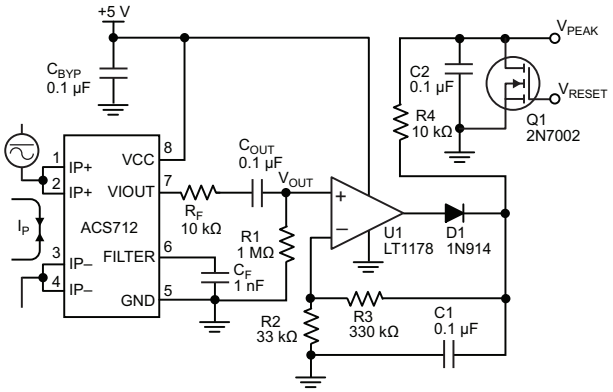
This technique is made possible through the use of a BiCMOS process that allows the use of low-offset and low-noise amplifiers in combination with high-density logic integration and sample and hold circuits.



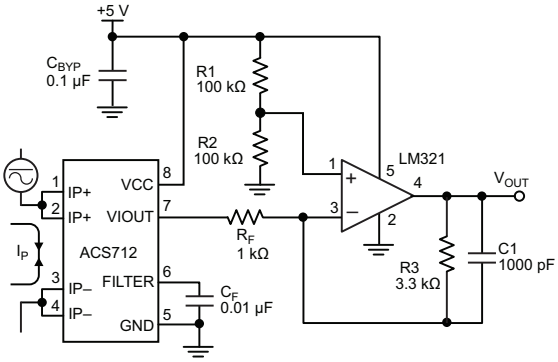
Concept of Chopper Stabilization Technique

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

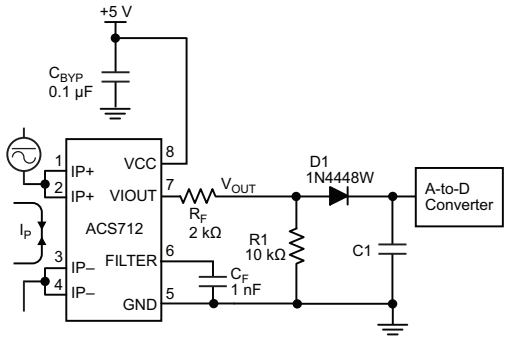
TYPICAL APPLICATIONS



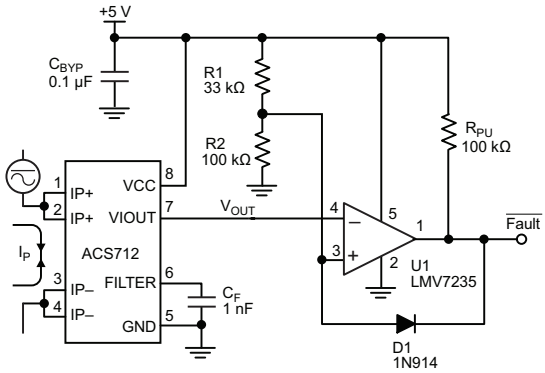
Application 2. Peak Detecting Circuit



Application 3. This configuration increases gain to 610 mV/A (tested using the ACS712ELC-05A).



Application 4. Rectified Output. 3.3 V scaling and rectification application for A-to-D converters. Replaces current transformer solutions with simpler ACS circuit. C1 is a function of the load resistance and filtering desired. R1 can be omitted if the full range is desired.



Application 5. 10 A Overcurrent Fault Latch. Fault threshold set by R1 and R2. This circuit latches an overcurrent fault and holds it until the 5 V rail is powered down.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

IMPROVING SENSING SYSTEM ACCURACY USING THE FILTER PIN

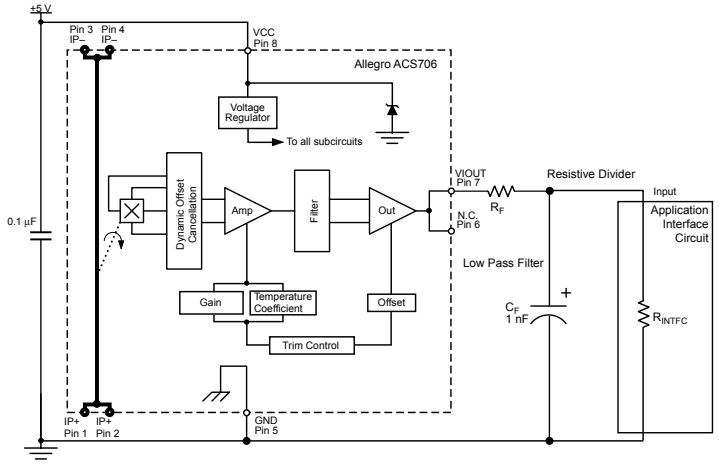
In low-frequency sensing applications, it is often advantageous to add a simple RC filter to the output of the device. Such a low-pass filter improves the signal-to-noise ratio, and therefore the resolution, of the device output signal. However, the addition of an RC filter to the output of a sensor IC can result in undesirable device output attenuation — even for DC signals.

Signal attenuation, ΔV_{ATT}, is a result of the resistive divider effect between the resistance of the external filter, R_F (see Application 6), and the input impedance and resistance of the customer interface circuit, R_{INTFC}. The transfer function of this resistive divider is given by:

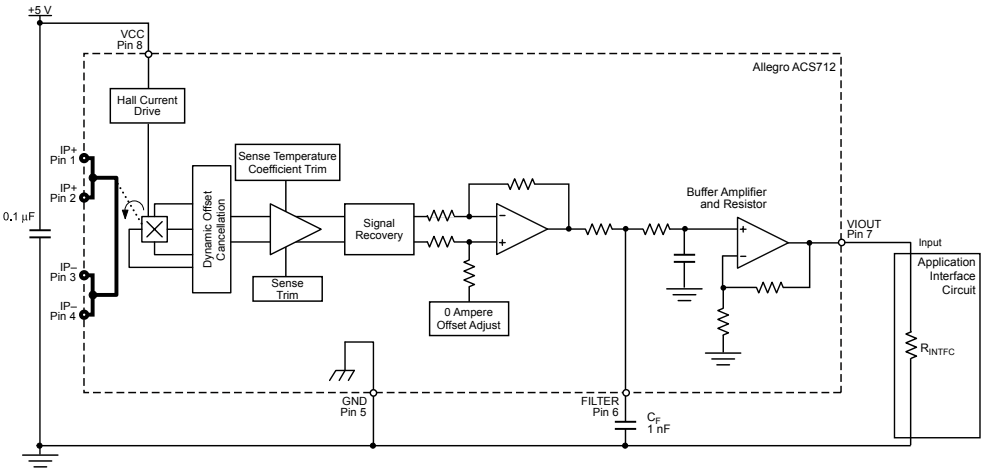
$$\Delta V_{ATT} = V_{IOUT}\left(\frac{R_{INTFC}}{R_F + R_{INTFC}}\right) .$$

Even if R_F and R_{INTFC} are designed to match, the two individual resistance values will most likely drift by different amounts over temperature. Therefore, signal attenuation will vary as a function of temperature. Note that, in many cases, the input impedance, R_{INTFC}, of a typical analog-to-digital converter (ADC) can be as low as 10 kΩ.

The ACS712 contains an internal resistor, a FILTER pin connection to the printed circuit board, and an internal buffer amplifier. With this circuit architecture, users can implement a simple RC filter via the addition of a capacitor, C_F (see Application 7) from the FILTER pin to ground. The buffer amplifier inside of the ACS712 (located after the internal resistor and FILTER pin connection) eliminates the attenuation caused by the resistive divider effect described in the equation for ΔV_{ATT}. Therefore, the ACS712 device is ideal for use in high-accuracy applications that cannot afford the signal attenuation associated with the use of an external RC low-pass filter.



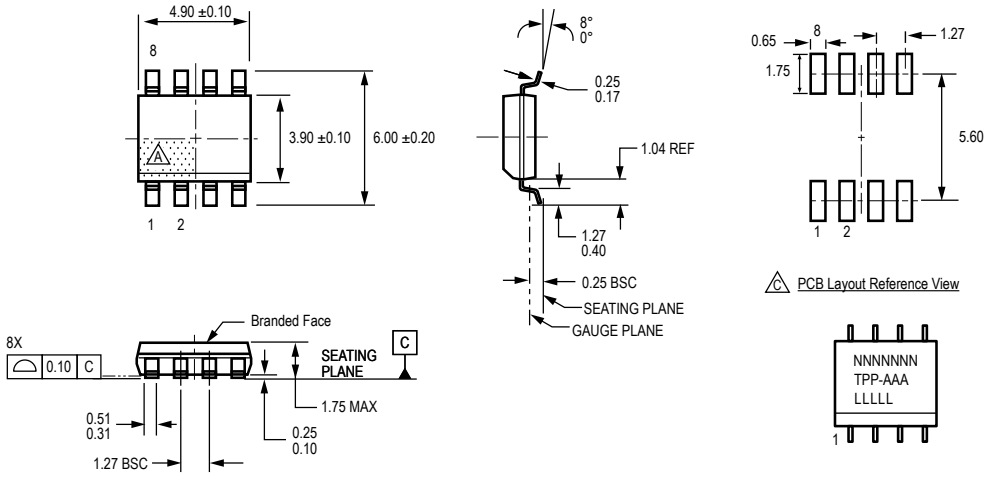
Application 6. When a low pass filter is constructed externally to a standard Hall effect device, a resistive divider may exist between the filter resistor, R_F, and the resistance of the customer interface circuit, R_{INTFC}. This resistive divider will cause excessive attenuation, as given by the transfer function for ΔV_{ATT}.



Application 7. Using the FILTER pin provided on the ACS712 eliminates the attenuation effects of the resistor divider between R_F and R_{INTFC}, shown in Application 6.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Package LC, 8-pin SOIC



For Reference Only; not for tooling use (reference MS-012AA)
Dimensions in millimeters
Dimensions exclusive of mold flash, gate burrs, and dambar protrusions
Exact case and lead configuration at supplier discretion within limits shown

A Terminal #1 mark area

B Branding scale and appearance at supplier discretion

C Reference land pattern layout (reference IPC7351 SOIC127P600X175-8M); all pads a minimum of 0.20 mm from all adjacent pads; adjust as necessary to meet application process requirements and PCB layout tolerances

N = Device part number
T = Device temperature range
P = Package Designator
A = Amperage
L = Lot number
Belly Brand = Country of Origin

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

REVISION HISTORY

Number	Date	Description
15	November 16, 2012	Update rise time and isolation, I_{OUT} reference data, patents
16	June 5, 2017	Updated product status
17	December 10, 2018	Updated certificate numbers
18	May 17, 2019	Updated TUV certificate mark, and minor editorial updates
19	January 30, 2020	Updated product status and minor editorial updates

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com