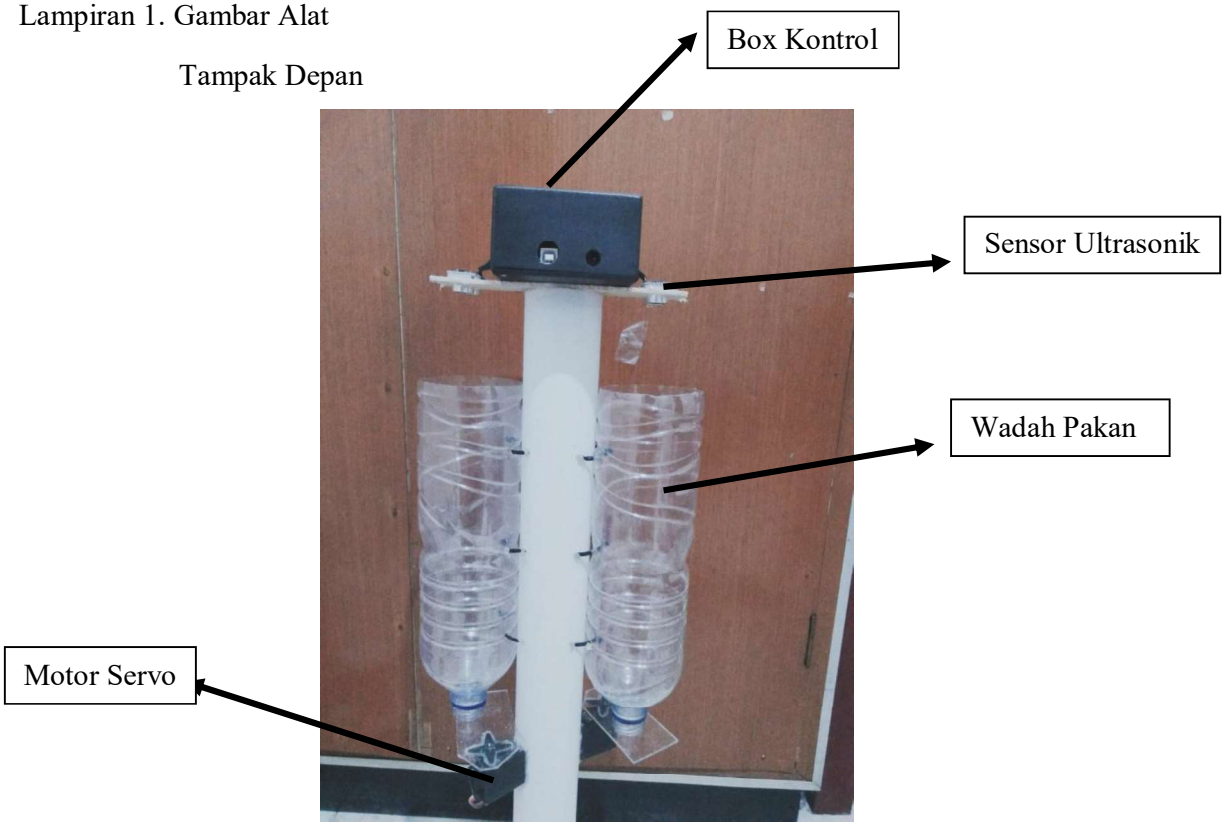
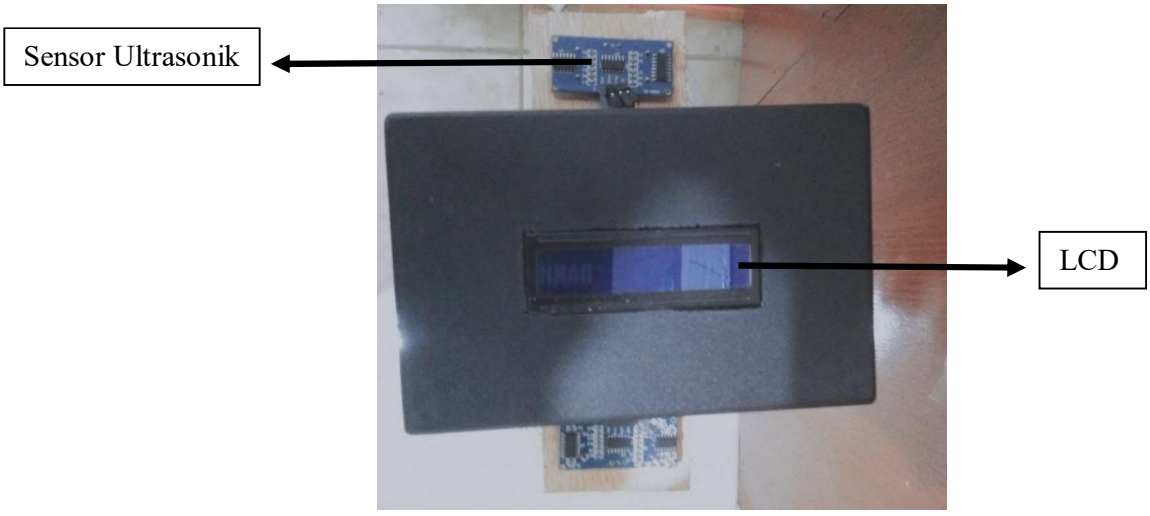
LAMPIRAN

Lampiran 1. Gambar Alat

Tampak Depan



Tampak Atas



Lampiran 2. Koding Keseluruhan

```
#include <RTClib.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <SoftwareSerial.h>

SoftwareSerial SIM800L (4,5); //rx|tx
Servo pakanSatu;
Servo pakanDua;
RTC_DS1307 rtc;
LiquidCrystal_I2C lcd (0x27,2,1,0,4,5,6,7,3,POSITIVE);

char namahari[7][12] = {"Minggu","Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"};
int pos = 0;
int trigSatu = 9;
int echoSatu = 8;
int trigDua = 11;
int echoDua = 10;
float durasiSatu, jarakSatu, jarakSatuu;
float durasiDua, jarakDua, jarakDuaa;
float mj = 0.65; //massa jenis
int beratSatu;
int beratDua;
int satu, dua;
float volumeSatu;
float volumeDua;
String pesanSMS1, pesanSMS2, pesanSMS3, pesanSMS4, pesanSMS5, pesanSMS6,
pesanSMS7, pesanSMS8, pesanSMS9, pesanSMS10, pesanSMS11;

void sensorUs(){
  digitalWrite (trigSatu, LOW);
  delayMicroseconds (8);
```

```
  digitalWrite (trigSatu, HIGH);
  delayMicroseconds (8);
  digitalWrite (trigSatu, LOW);
  delayMicroseconds (8);
  durasiSatu = pulseIn (echoSatu, HIGH);
  jarakSatu = (durasiSatu*0.034/2);
  satu = (jarakSatu - 5);
  jarakSatuu = (26-jarakSatu); //jika p maks 21
  volumeSatu = ((22*4*4*jarakSatuu)/7); //volume tabung
  beratSatu =(mj*volumeSatu);

  digitalWrite (trigDua, LOW);
  delayMicroseconds (8);
  digitalWrite (trigDua, HIGH);
  delayMicroseconds (8);
  digitalWrite (trigDua, LOW);
  delayMicroseconds (8);
  durasiDua = pulseIn (echoDua, HIGH);
  jarakDua = (durasiDua*0.034/2);
  dua = (jarakDua - 6);
  jarakDuaa = (26-dua); //jika p maks 21
  volumeDua = ((22*4*4*jarakDuaa)/7); //volume tabung
  beratDua =(mj*volumeDua);
}

void kirim (){
  String pesanSMS1 = ("Pemberian Pakan Telah dilaksanakan dan sisa pakan yang
tersedia: ");
  String pesanSMS2 = ("Massa1 = ")+String(beratSatu)+(" gram ");
  String pesanSMS3 = ("Massa2 = ")+String(beratDua)+(" gram ");
  String pesanSMS4 = ("Peringatan Pakan Akan Segera Habis (1)");
  String pesanSMS5 = ("Peringatan Pakan Akan Segera Habis (1) dan Pakan Habis (2)");
  String pesanSMS6 = ("Peringatan Pakan Akan Segera Habis (2)");
  String pesanSMS7 = ("Pakan Habis (1) dan Peringatan Pakan Akan Segera Habis (2)");
```

```
String pesanSMS8 = ("Peringatan Pakan Akan Segera Habis (1) & (2)");
String pesanSMS9 = ("Pakan Habis (1)");
String pesanSMS10 = ("Pakan Habis (2)");
String pesanSMS11 = ("Pakan Habis (1) & (2)");

SIM800L.print("AT+CMGF=1\r\n");
delay(1000);
SIM800L.print("AT+CMGS=\"+6285714371509\"\r\n");
delay (1000);

if (beratSatu >= 500 && beratDua >= 500){
 SIM800L.print(pesanSMS1);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS2);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS3);
 SIM800L.println(" ");
 SIM800L.write((char)26);
 delay(1000);
}
if (beratSatu >= 250 && beratSatu < 500 && beratDua >= 500){
 SIM800L.print(pesanSMS1);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS2);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS3);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS4);
 SIM800L.println(" ");
 SIM800L.write((char)26);
 delay(1000);
}
if (beratSatu >= 250 && beratSatu < 500 && beratDua < 250){
 SIM800L.print(pesanSMS1);
```

```
  SIM800L.println(" ");
  SIM800L.print(pesanSMS2);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS3);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS5);
  SIM800L.println(" ");
  SIM800L.write((char)26);
  delay(1000);
 }
 if (beratDua >= 250 && beratDua < 500 && beratSatu >= 500){
  SIM800L.print(pesanSMS1);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS2);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS3);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS6);
  SIM800L.println(" ");
  SIM800L.write((char)26);
  delay(1000);
 }
 if (beratSatu < 250 && beratDua >=250 && beratDua <500 ){
  SIM800L.print(pesanSMS1);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS2);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS3);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS7);
  SIM800L.println(" ");
  SIM800L.write((char)26);
  delay(1000);
 }
```

```
if (beratSatu >= 250 && beratSatu >500 && beratDua >=250 && beratDua <500 ){
 SIM800L.print(pesanSMS1);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS2);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS3);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS8);
 SIM800L.println(" ");
 SIM800L.write((char)26);
 delay(1000);
}
if (beratSatu < 250 && beratDua >=500){
 SIM800L.print(pesanSMS1);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS2);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS3);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS9);
 SIM800L.println(" ");
 SIM800L.write((char)26);
 delay(1000);
}
if (beratDua < 250 && beratSatu >=500){
 SIM800L.print(pesanSMS1);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS2);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS3);
 SIM800L.println(" ");
 SIM800L.print(pesanSMS10);
 SIM800L.println(" ");
 SIM800L.write((char)26);
```

```
  delay(1000);
 }
 if (beratSatu < 250 && beratDua < 250){
  SIM800L.print(pesanSMS1);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS2);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS3);
  SIM800L.println(" ");
  SIM800L.print(pesanSMS11);
  SIM800L.println(" ");
  SIM800L.write((char)26);
  delay(1000);
  }
 }
void setup() {
 Serial.begin (9600);
 SIM800L.begin (9600);
 lcd.begin(16,2);

 pinMode (trigSatu, OUTPUT);
 pinMode (echoSatu, INPUT);
 pinMode (trigDua, OUTPUT);
 pinMode (echoDua, INPUT);

 if (! rtc.begin()){
  Serial.println("Couldn't find RTC");
  while (1);
 }
 if (! rtc.isrunning()){
  Serial.println("RTC is NOT running!");
  rtc.adjust(DateTime(2019,9,20,18,2,00));
 }
 pakanSatu.attach(6);
```

```
  pakanDua.attach(7);
  pakanSatu.write(90);
  pakanDua.write(90);
}

void loop(){
  DateTime now = rtc.now();
  lcd.setCursor(0,0);
  lcd.print (namahari [now.dayOfTheWeek()]);
  lcd.print (',');
  lcd.print (now.day(), DEC);
  lcd.print('-');
  lcd.print(now.month(), DEC);
  lcd.print('-');
  lcd.print(now.year(), DEC);

  lcd.setCursor(8,1);
  lcd.print(now.hour(), DEC);
  lcd.print(':');
  lcd.print(now.minute(), DEC);
  lcd.print(':');
  lcd.print(now.second(), DEC);
  delay (1000);

  if (now.hour() == 6 && now.minute() == 30&& now.second() == 00){
   for (pos = 0; pos <= 2; pos += 1){
     pakanSatu.write(180);
     pakanDua.write (180);
     delay(1000);
     pakanSatu.write(90);
     pakanDua.write (90);
     delay(1000);
     Serial.println ("Pakan Terbuka Pagi");
   }
```

```
}
if (now.hour() == 6 && now.minute() == 30&& now.second() == 30){
 sensorUs();
 lcd.clear();
 lcd.setCursor (0,0);
 lcd.print ("Massa1 = ");
 lcd.print (beratSatu);
 lcd.print (" g");
 lcd.setCursor (0,1);
 lcd.print ("Massa2 = ");
 lcd.print (beratDua);
 lcd.print (" g");
 delay (3000);
 kirim();
}

if (now.hour() == 16 && now.minute () == 00&& now.second () == 00) {
 for (pos = 0; pos <= 2; pos += 1){
   pakanSatu.write(180);
   pakanDua.write (180);
   delay(1000);
   pakanSatu.write(90);
   pakanDua.write (90);
   delay(1000);
   Serial.println ("Pakan Terbuka Sore");
  }
}
 if (now.hour() == 16 && now.minute() == 00 && now.second() == 30){
   sensorUs();
   lcd.clear();
   lcd.setCursor (0,0);
   lcd.print ("Massa1 = ");
   lcd.print (beratSatu);
   lcd.print (" g");
```

```
    lcd.setCursor (0,1);
    lcd.print ("Massa2 = ");
    lcd.print (beratDua);
    lcd.print (" g");
    delay (3000);
    kirim();
   }
  lcd.clear ();
}
```

Lampiran 3. Datasheet RTC DS1307

maxim integrated™

DS1307
64 x 8, Serial, I²C Real-Time Clock

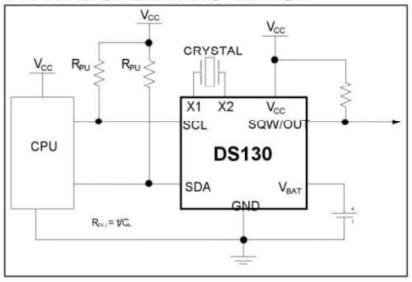
GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

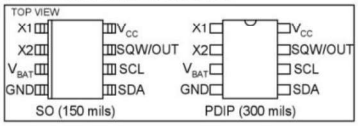
FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Compensation Valid Up to 2100
- 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratories (UL) Recognized

TYPICAL OPERATING CIRCUIT



PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.

REV: 100208

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground ..... -0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial ..... 0°C to +70°C
Industrial ..... -40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads) ..... +260°C for 10 seconds
Soldering Temperature (surface mount) ..... Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}		0		100	kHz
Bus Free Time Between a STOP and START Condition	t_{BUF}		4.7			µs
Hold Time (Repeated) START Condition	$t_{HD:STA}$	(Note 4)	4.0			µs
LOW Period of SCL Clock	t_{LOW}		4.7			µs
HIGH Period of SCL Clock	t_{HIGH}		4.0			µs
Setup Time for a Repeated START Condition	$t_{SU:STA}$		4.7			µs
Data Hold Time	$t_{HD:DAT}$		0			µs
Data Setup Time	$t_{SU:DAT}$	(Notes 5, 6)	250			ns
Rise Time of Both SDA and SCL Signals	t_R				1000	ns
Fall Time of Both SDA and SCL Signals	t_F				300	ns
Setup Time for STOP Condition	$t_{SU:STO}$		4.7			µs

CAPACITANCE

(T_A = +25°C)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Pin Capacitance (SDA, SCL)	$C_{I/O}$				10	pF
Capacitance Load for Each Bus Line	C_B	(Note 7)			400	pF

Note 1: All voltages are referenced to ground.

Note 2: Limits at -40°C are guaranteed by design and are not production tested.

Note 3: I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.

Note 4: After this period, the first clock pulse is generated.

Note 5: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

Note 6: The maximum $t_{HD:DAT}$ only has to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

Note 7: C_B—total capacitance of one bus line in pF.

TIMING DIAGRAM

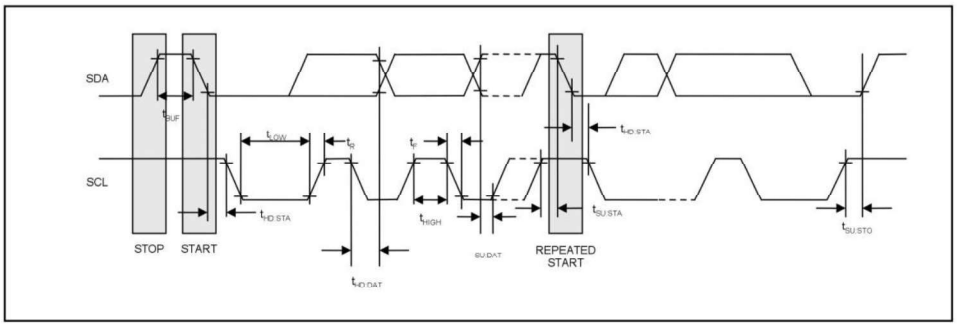
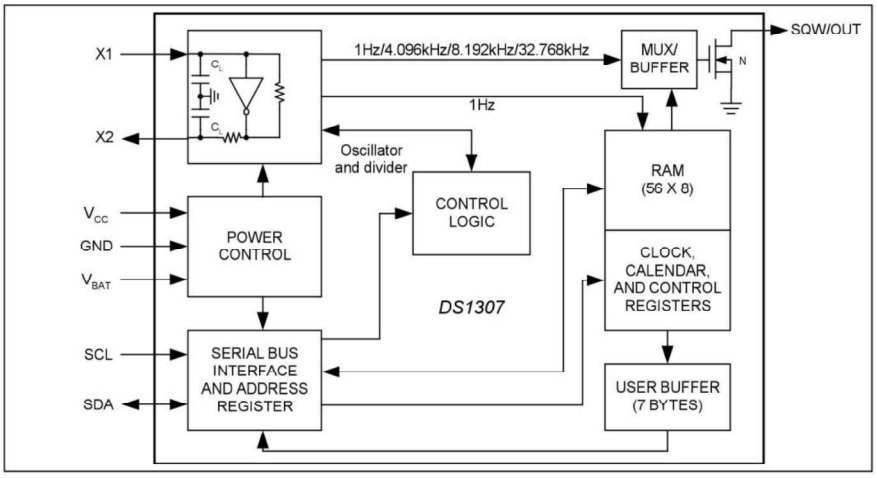
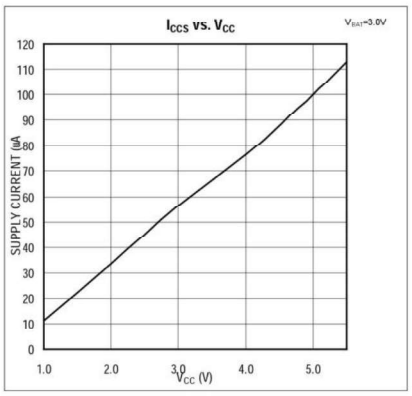


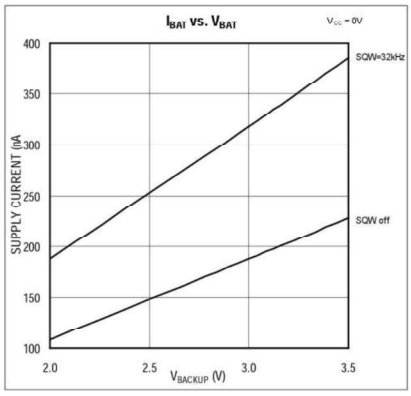
Figure 1. Block Diagram

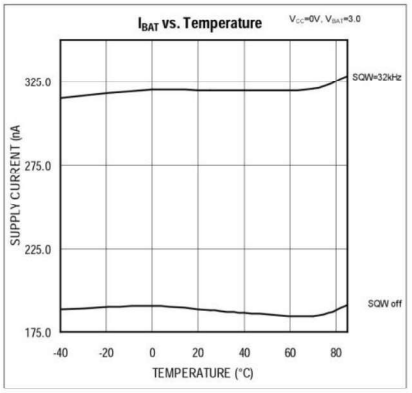


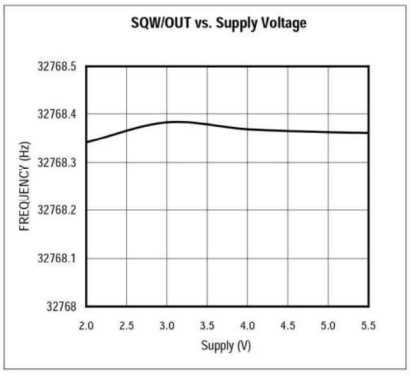
TYPICAL OPERATING CHARACTERISTICS

(V_{CC} = 5.0V, T_A = +25°C, unless otherwise noted.)









PIN DESCRIPTION

PIN	NAME	FUNCTION
1	X1	Connections for Standard 32.768kHz Quartz Crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (C_L) of 12.5pF. X1 is the input to the oscillator and can optionally be connected to an external 32.768kHz oscillator. The output of the internal oscillator, X2, is floated if an external oscillator is connected to X1. **Note:** For more information on crystal selection and crystal layout considerations, refer to *Application Note 58: Crystal Considerations with Dallas Real-Time Clocks*.
2	X2	
3	V_{BAT}	Backup Supply Input for Any Standard 3V Lithium Cell or Other Energy Source. Battery voltage must be held between the minimum and maximum limits for proper operation. Diodes in series between the battery and the V_{BAT} pin may prevent proper operation. If a backup supply is not required, V_{BAT} must be grounded. The nominal power-fail trip point (V_{PF}) voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAh or greater will back up the DS1307 for more than 10 years in the absence of power at +25°C. UL recognized to ensure against reverse charging current when used with a lithium battery. Go to: www.maxim-ic.com/qa/info/ul/.
4	GND	Ground
5	SDA	Serial Data Input/Output. SDA is the data input/output for the I²C serial interface. The SDA pin is open drain and requires an external pullup resistor. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
6	SCL	Serial Clock Input. SCL is the clock input for the I²C interface and is used to synchronize data movement on the serial interface. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
7	SQW/OUT	Square Wave/Output Driver. When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square-wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pullup resistor. SQW/OUT operates with either V_{CC} or V_{BAT} applied. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}. If not used, this pin can be left floating.
8	V_{CC}	Primary Power Supply. When voltage is applied within normal limits, the device is fully accessible and data can be written and read. When a backup supply is connected to the device and V_{CC} is below V_{TP}, read and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage.

DETAILED DESCRIPTION

The DS1307 is a low-power clock/calendar with 56 bytes of battery-backed SRAM. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The DS1307 operates as a slave device on the I²C bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT}, the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out-of-tolerance system. When V_{CC} falls below V_{BAT}, the device switches into a low-current battery-backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} +0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

OSCILLATOR CIRCUIT

The DS1307 uses an external 32.768kHz crystal. The oscillator circuit does not require any external resistors or capacitors to operate. Table 1 specifies several crystal parameters for the external crystal. Figure 1 shows a functional schematic of the oscillator circuit. If using a crystal with the specified characteristics, the startup time is usually less than one second.

CLOCK ACCURACY

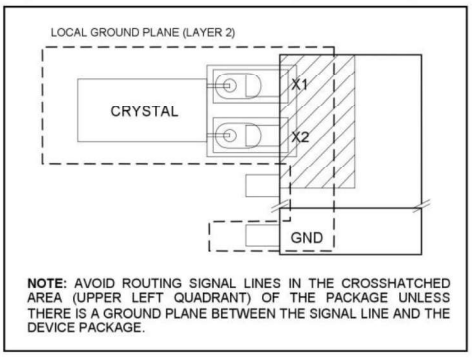
The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. Refer to Application Note 58: *Crystal Considerations with Dallas Real-Time Clocks* for detailed information.

Table 1. Crystal Specifications*

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
Nominal Frequency	f_O		32.768		kHz
Series Resistance	ESR			45	kΩ
Load Capacitance	C_L		12.5		pF

**The crystal, traces, and crystal input pins should be isolated from RF generating signals. Refer to* Application Note 58: Crystal Considerations for Dallas Real-Time Clocks *for additional specifications.*

Figure 2. Recommended Layout for Crystal



RTC AND RAM ADDRESS MAP

Table 2 shows the address map for the DS1307 RTC and RAM registers. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multibyte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

CLOCK AND CALENDAR

The time and calendar information is obtained by reading the appropriate register bytes. Table 2 shows the RTC registers. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. The day-of-week register increments at midnight. Values that correspond to the day of week are user-defined but must be sequential (i.e., if 1 equals Sunday, then 2 equals Monday, and so on.) Illogical time and date entries result in undefined operation. Bit 7 of Register 0 is the clock halt (CH) bit. When this bit is set to 1, the oscillator is disabled. When cleared to 0, the oscillator is enabled. On first application of power to the device the time and date registers are typically reset to 01/01/00 01 00:00:00 (MM/DD/YY DOW HH:MM:SS). The CH bit in the seconds register will be set to a 1. The clock can be halted whenever the timekeeping functions are not required, which minimizes current (I_{BATDR}).

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12-hour or 24-hour mode-select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to 23 hours). The hours value must be re-entered whenever the 12/24-hour mode bit is changed.

When reading or writing the time and date registers, secondary (user) buffers are used to prevent errors when the internal registers update. When reading the time and date registers, the user buffers are synchronized to the internal registers on any I^2C START. The time information is read from these secondary registers while the clock continues to run. This eliminates the need to re-read the registers in case the internal registers update during a read. The divider chain is reset whenever the seconds register is written. Write transfers occur on the I^2C acknowledge from the DS1307. Once the divider chain is reset, to avoid rollover issues, the remaining time and date registers must be written within one second.

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

CONTROL REGISTER

The DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

Bit 7: Output Control (OUT). This bit controls the output level of the SQW/OUT pin when the square-wave output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0. On initial application of power to the device, this bit is typically set to a 0.

Bit 4: Square-Wave Enable (SQWE). This bit, when set to logic 1, enables the oscillator output. The frequency of the square-wave output depends upon the value of the RS0 and RS1 bits. With the square-wave output set to 1Hz, the clock registers update on the falling edge of the square wave. On initial application of power to the device, this bit is typically set to a 0.

Bits 1 and 0: Rate Select (RS[1:0]). These bits control the frequency of the square-wave output when the square-wave output has been enabled. The following table lists the square-wave frequencies that can be selected with the RS bits. On initial application of power to the device, these bits are typically set to a 1.

RS1	RS0	SQW/OUT OUTPUT	SQWE	OUT
0	0	1Hz	1	X
0	1	4.096kHz	1	X
1	0	8.192kHz	1	X
1	1	32.768kHz	1	X
X	X	0	0	0
X	X	1	0	1

I²C DATA BUS

The DS1307 supports the I²C protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307 operates as a slave on the I²C bus.

Figures 3, 4, and 5 detail how data is transferred on the I²C bus.

- Data transfer can be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

START data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

STOP data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

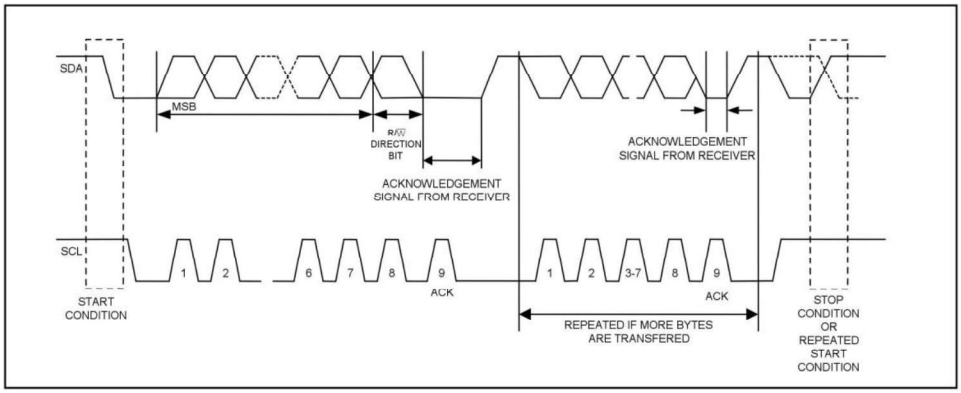
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the I²C bus specifications a standard mode (100kHz clock rate) and a fast mode (400kHz clock rate) are defined. The DS1307 operates in the standard mode (100kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

Figure 3. Data Transfer on I^2C Serial Bus



Depending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

   The master device generates all the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS1307 can operate in the following two modes:

1. **Slave Receiver Mode (Write Mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Hardware performs address recognition after reception of the slave address and direction bit (see Figure 4). The slave address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which for a write is 0. After receiving and decoding the slave address byte, the DS1307 outputs an acknowledge on SDA. After the DS1307 acknowledges the slave address + write bit, the master transmits a word address to the DS1307. This sets the register pointer on the DS1307, with the DS1307 acknowledging the transfer. The master can then transmit zero or more bytes of data with the DS1307 acknowledging each byte received. The register pointer automatically increments after each data byte are written. The master will generate a STOP condition to terminate the data write.

2. **Slave Transmitter Mode (Read Mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. The DS1307 transmits serial data on SDA while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (see Figure 5). The slave address byte is the first byte received after the START condition is generated by the master. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which is 1 for a read. After receiving and decoding the slave address the DS1307 outputs an acknowledge on SDA. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The register pointer automatically increments after each byte are read. The DS1307 must receive a Not Acknowledge to end a read.

Figure 4. Data Write—Slave Receiver Mode

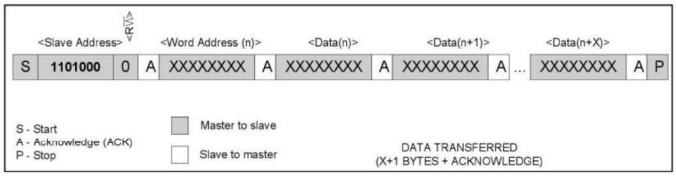


Figure 5. Data Read—Slave Transmitter Mode

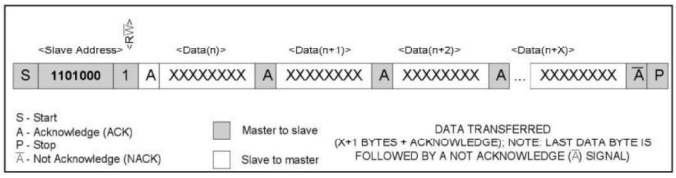
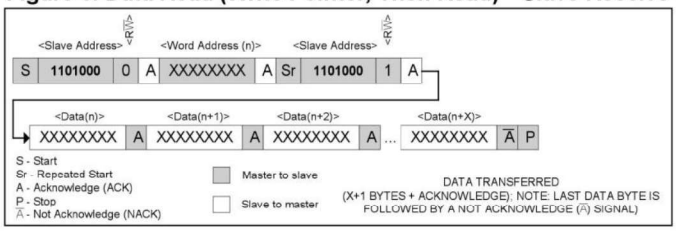


Figure 6. Data Read (Write Pointer, Then Read)—Slave Receive and Transmit



PACKAGE INFORMATION

For the latest package outline information and land patterns, go to **www.maxim-ic.com/packages**.

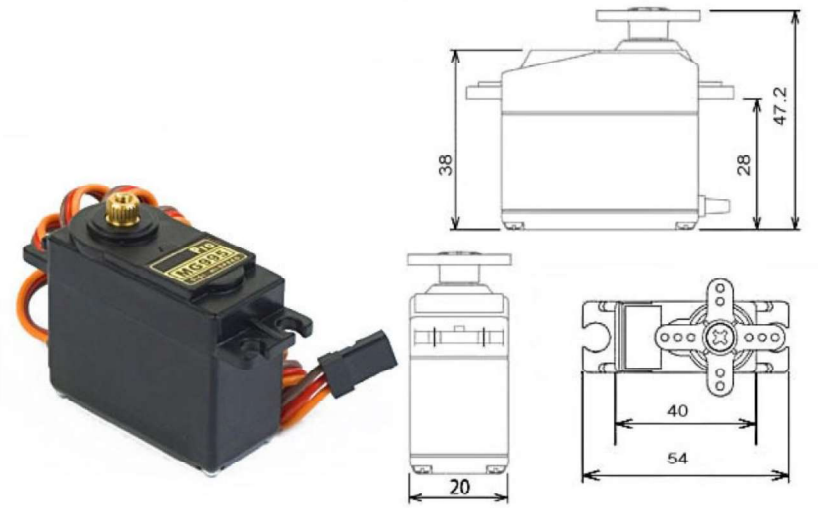
PACKAGE TYPE	PACKAGE CODE	DOCUMENT NO.
8 PDIP	—	21-0043
8 SO	—	21-0041

REVISION HISTORY

REVISION DATE	DESCRIPTION	PAGES CHANGED
100208	Moved the *Typical Operating Circuit* and *Pin Configurations* to first page.	1
	Removed the leaded part numbers from the *Ordering Information* table.	1
	Added an open-drain transistor to SQW/OUT in the block diagram (Figure 1).	4
	Added the pullup voltage range for SDA, SCL, and SQW/OUT to the *Pin Description* table and noted that SQW/OUT can be left open if not used.	6
	Added default time and date values on first application of power to the *Clock and Calendar* section and deleted the note that initial power-on state is not defined.	8
	Added default on initial application of power to bit info in the *Control Register* section.	9
	Updated the *Package Information* section to reflect new package outline drawing numbers.	13

Lampiran 4. Datasheet Motor Servo

MG995 High Speed
Metal Gear Dual Ball Bearing Servo

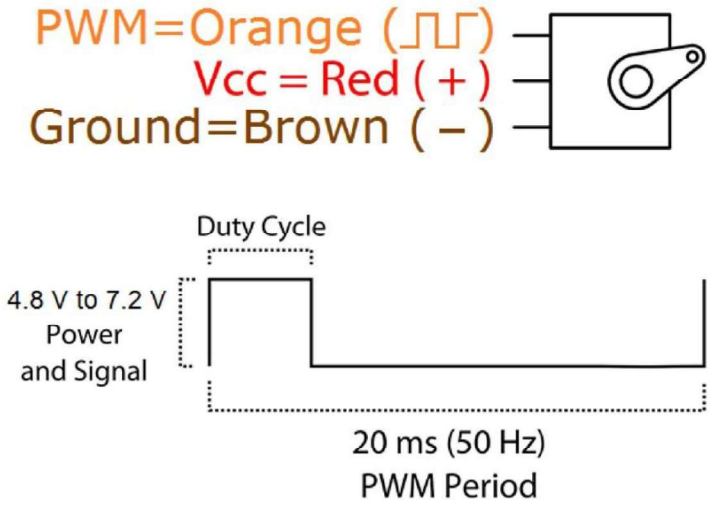


The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-speed standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG995 Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC


PWM=Orange
Vcc = Red (+)
Ground=Brown (–)
Duty Cycle
4.8 V to 7.2 V
Power
and Signal
20 ms (50 Hz)
PWM Period

Lampiran 5. Datasheet Ultrasonik

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

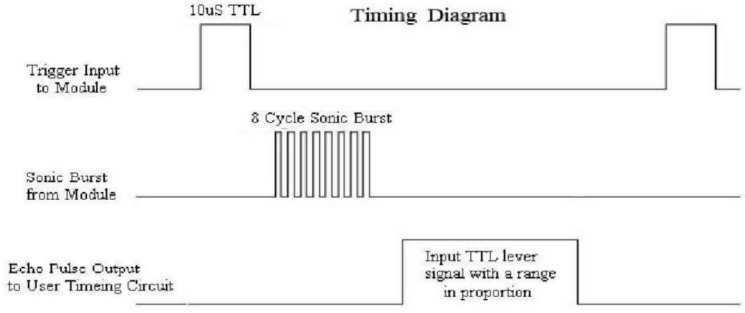
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com

Lampiran 6. Datasheet SIM800L

1. Introduction

This document describes SIM800L hardware interface in great detail.
This document can help user to quickly understand SIM800L interface specifications, electrical and mechanical details. With the help of this document and other SIM800L application notes, user guide, users can use SIM800L to design various applications quickly.

2. SIM800L Overview

SIM800L is a quad-band GSM/GPRS module, that works on frequencies GSM850MHz, EGSM900MHz, DCS1800MHz and PCS1900MHz. SIM800L features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

With a tiny configuration of 15.8*17.8*2.4mm, SIM800L can meet almost all the space requirements in user applications, such as smart phone, PDA and other mobile devices.

SIM800L has 88pin pads of LGA packaging, and provides all hardware interfaces between the module and customers' boards.

- Support 5*5*2 keypads
- One full modem serial port, user can configure two serial ports
- One USB, the USB interfaces can debug, download software
- Audio channel which includes two microphone input; a receiver output and a speaker output
- Programmable general purpose input and output.
- A SIM card interface
- Support FM
- Support one PWM

SIM800L is designed with power saving technique so that the current consumption is as low as 0.7mA in sleep mode.

2.1. SIM800L Key Features

Table 1: SIM800L key features

Feature	Implementation
Power supply	3.4V ~4.4V
Power saving	typical power consumption in sleep mode is 0.7mA (AT+CFUN=0)
Frequency bands	● Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM800L can search the 4 frequency bands automatically. The frequency bands can also be set by AT command "AT+CBAND". For details, please refer to document [1]. ● Compliant to GSM Phase 2/2+
Transmitting power	● Class 4 (2W) at GSM 850 and EGSM 900 ● Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	● GPRS multi-slot class 12（default） ● GPRS multi-slot class 1~12 (option)
Temperature range	● Normal operation: -40°C ~ +85°C

	● Storage temperature -45°C ~ +90°C
Data GPRS	● GPRS data downlink transfer: max. 85.6 kbps ● GPRS data uplink transfer: max. 85.6 kbps ● Coding scheme: CS-1, CS-2, CS-3 and CS-4 ● PAP protocol for PPP connect ● Integrate the TCP/IP protocol. ● Support Packet Broadcast Control Channel (PBCCH) ● CSD transmission rates：2.4，4.8，9.6，14.4 kbps
CSD	● Support CSD transmission
USSD	● Unstructured Supplementary Services Data (USSD) support
SMS	● MT, MO, CB, Text and PDU mode ● SMS storage: SIM card
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	Speech codec modes: ● Half Rate (ETS 06.20) ● Full Rate (ETS 06.10) ● Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80) ● Adaptive multi rate (AMR) ● Echo Cancellation ● Noise Suppression
Serial port and debug port	**Serial port:** ● Full modem interface with status and control lines, unbalanced, asynchronous. ● 1200bps to 115200bps. ● Can be used for AT commands or data stream. ● Support RTS/CTS hardware handshake and software ON/OFF flow control. ● Multiplex ability according to GSM 07.10 Multiplexer Protocol. ● Autobauding supports baud rate from 1200 bps to 57600bps. ● upgrading firmware **Debug port:** ● USB_DM and USB_DP ● Can be used for debugging and upgrading firmware.
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC.
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Timing functions	Use AT command set
Physical characteristics	Size:15.8*17.8*2.4mm Weight:1.35g
Firmware upgrade	Main serial port or USB port.

Table 2: Coding schemes and maximum net data rates over air interface

Coding scheme	1 timeslot	2 timeslot	4 timeslot
CS-1	9.05kbps	18.1kbps	36.2kbps
CS-2	13.4kbps	26.8kbps	53.6kbps
CS-3	15.6kbps	31.2kbps	62.4kbps
CS-4	21.4kbps	42.8kbps	85.6kbps

2.2. Operating Mode

The table below summarizes the various operating modes of SIM800L.

Table 3: Overview of operating modes

Mode	Function	
Normal operation	GSM/GPRS SLEEP	Module will automatically go into sleep mode if the conditions of sleep mode are enabling and there is no on air and no hardware interrupt (such as GPIO interrupt or data on serial port). In this case, the current consumption of module will reduce to the minimal level. In sleep mode, the module can still receive paging message and SMS.
	GSM IDLE	Software is active. Module is registered to the GSM network, and the module is ready to communicate.
	GSM TALK	Connection between two subscribers is in progress. In this case, the power consumption depends on network settings such as DTX off/on, FR/EFR/HR, hopping sequences, antenna.
	GPRS STANDBY	Module is ready for GPRS data transfer, but no data is currently sent or received. In this case, power consumption depends on network settings and GPRS configuration.
	GPRS DATA	There is GPRS data transfer (PPP or TCP or UDP) in progress. In this case, power consumption is related with network settings (e.g. power control level); uplink/downlink data rates and GPRS configuration (e.g. used multi-slot settings).
Power down	Normal power down by sending AT command “AT+CPOWD=1” or using the PWRKEY. The power management unit shuts down the power supply for the baseband part of the module, and only the power supply for the RTC is remained. Software is not active. The serial port is not accessible. Power supply (connected to VBAT) remains applied.	
Minimum functionality mode	AT command “AT+CFUN” can be used to set the module to a minimum functionality mode without removing the power supply. In this mode, the RF part of the module will not work or the SIM card will not be accessible, or both RF part and SIM card will be closed, and the serial port is still accessible. The power consumption in this mode is lower than normal mode.	

2.3. Functional Diagram

The following figure shows a functional diagram of SIM800L:

- GSM baseband
- GSM RF
- Antenna interface
- Other interface

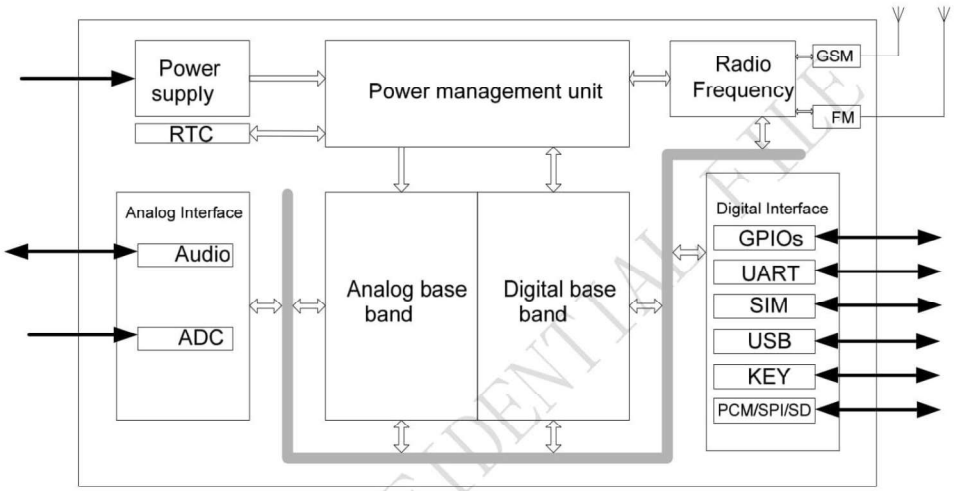


Figure 1: SIM800L functional diagram

3.2. Pin Description

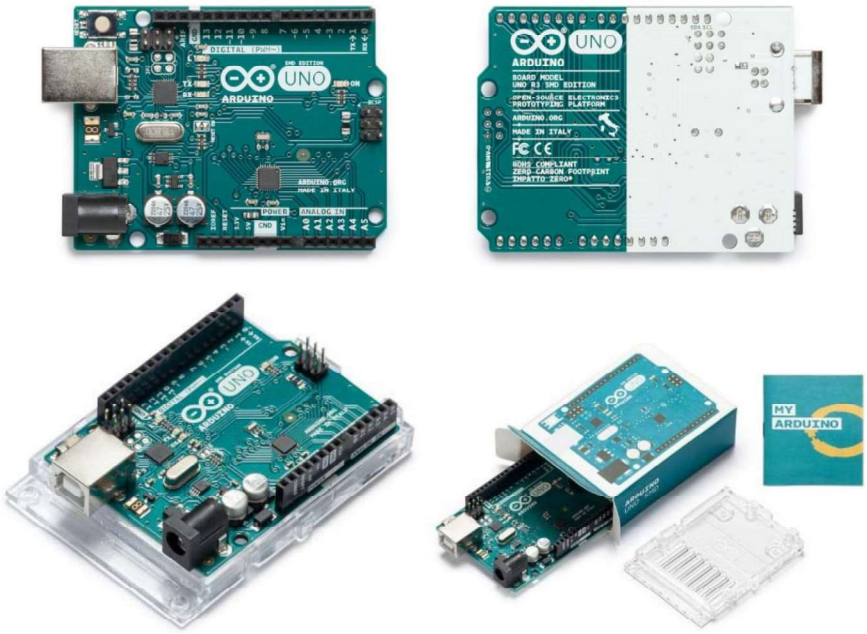
Table 4: Pin description

Pin name	Pin number	I/O	Description	Comment
Power supply				
VBAT	1,42	I	Power supply	
VRTC	56	I/O	Power supply for RTC	It is recommended to connect with a battery or a capacitor (e.g. 4.7uF).
VEXT	18	O	2.8V power output	If these pins are unused, keep open.
GND	2,6,8,35,37,38,39,41,43,44,45,58,67,71,72,73,76,77,78,79,80,81,82,83,84,85,86,87,88		Ground	GND for VBAT recommend to use 2,43,44,45pin
Power on/down				
PWRKEY	48	I	PWRKEY should be pulled low at least 1 second and then released to power on/down the module.	Internally pulled up to VBAT.
Audio interfaces				
MIC1P	52	I	Differential audio input	If these pins are unused, keep open.
MIC1N	12			
SPK1P	53	O	Differential audio output	
SPK1N	13			
MIC2P	9	I	Differential audio input	
MIC2N	10			
SPK2P	51	O	Differential audio output	
SPK2N	11			
PCM interface				
PCMCLK	29	O	PCM interface for audio	If these pins are unused, keep open.
PCMOUT	30	O		
PCMSYNC	65	O		
PCMIN	66	I		
Keypads interface				
COL4	24	I	Support up to 50 buttons (5*5*2)	If these pins are unused, keep open.（Pin number 20 external cannot be pulled down）
COL3	21	I		
COL2	22	I		
COL1	25	I		
COL0	20	I		
ROW4	63	O		
ROW3	23	O		

ROW2	61	O		
ROW1	60	O		
ROW0	62	O		
GPIO				
GPIO1	3	I/O	Programmable general purpose input and output	
GPIO2	27	I/O		
GPIO3	28	I/O		
NETLIGHT	64	O	Network status	
STATUS	4	O	Power on status	
Serial port				
UART_DTR	69	I	Data terminal ready	If these pins are unused, keep open.
UART_RI	68	O	Ring indicator	
UART_DCD	70	O	Data carrier detect	
CTS	34	O	Request to send	
RTS	33	I	Clear to send	
TXD	32	O	Transmit data	
RXD	31	I	Receive data	
Debug interface				
VBUS	7	I	Debug and download	If these pins are unused, keep open.
USB_DP	59	I/O		
USB_DM	19	I/O		
ADC				
ADC	50	I	10bit general analog to digital converter	If these pins are unused, keep open.
PWM				
PWM	26	O	Pulse-width modulation	If these pins are unused, keep open.
I^2C				
SDA	75	I/O	I^2C serial bus data	Need external pulled up
SCL	74	O	I^2C serial bus clock	
SIM card interface				
VSIM	16	O	Voltage supply for SIM card. Support 1.8V or 3V SIM card	All signals of SIM interface should be protected against ESD with a TVS diode array.
SIM_DATA	14	I/O	SIM data input/output	
SIM_CLK	55	O	SIM clock	
SIM_RST	15	O	SIM reset	
SIMPRE	54	I	SIM card detection	Reservation function
Antenna interface				
ANT	40	I/O	Connect GSM antenna	
FM_ANT_P	17	I	Differential antenna for FM	
FM_ANT_N	57	I		
Synchronizing signal of RF				

BPI_BUS1	5	O	Synchronizing signal of RF	
Other				
RESET	49	I	Reset input(Active low)	
ISINK1	46	I	Drive keypad backlight	
ISINK0	47	I	Drive LCD backlight	
NC				
NC	36			

Lampiran 7. Datasheet Arduino Uno



ARDUINO UNO REV3 SMD

Code: A000073

The board everybody gets started with, based on the ATmega328 (SMD).

- The Arduino Uno SMD R3 is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

  The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip.

  Additional features coming with the R3 version are:

  - ATmega16U2 instead 8U2 as USB-to-Serial converter.
  - 1.0 pinout: added SDA and SCL pins for TWI communication placed near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board and the second one is a not connected pin, that is reserved for future purposes.
  - stronger RESET circuit.

  "Uno" means "One" in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

TECH SPECS

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

OSH: Schematics

Arduino Uno is open-source hardware! You can build your own board using the follwing files:

EAGLE FILES IN .ZIP

https://www.arduino.cc/en/uploads/Main/arduino-uno-smd-reference-design.zip

SCHEMATICS IN .PDF

https://www.arduino.cc/en/uploads/Main/arduino-uno-smd-schematic.pdf

Programming

The Arduino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then rese ing the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Warnings

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Differences with other boards

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Power

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

- IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.

PIN MAPPING ATmega328P https://www.arduino.cc/en/Hacking/PinMapping168

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows serial communication on any of the Uno's digital pins. https://www.arduino.cc/en/Reference/SoftwareSerial

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

Revisions

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

- Stronger RESET circuit.

- Atmega 16U2 replace the 8U2.

https://store.arduino.cc/usa/arduino-uno-smd-rev3 12-6-17