Developer Kit
Legal Stuff

• Stensat Group LLC assumes no responsibility and/or liability for the use of the kit and documentation.

• There is a 90 day warranty for the Develop kit against component defects. Damage caused by the user or owner is not covered.
  - Warranty does not cover such things as over tightening nuts on standoffs to the point of breaking off the standoff threads, breaking wires off the motors, causing shorts to damage components, powering the motor driver backwards, plugging the power input into an AC outlet, applying more than 9 volts to the power input, dropping the kit, kicking the kit, throwing the kit in fits of rage, unforeseen damage caused by the user/owner or any other method of destruction.

• If you do cause damage, we can sell you replacement parts or you can get most replacement parts from online hardware distributors.

• This document can be copied and printed and used by individuals who bought the kit, classroom use, summer camp use, and anywhere the kit is used. Stealing and using this document for profit is not allowed.

• If you need to contact us, go to www.stensat.org and click on contact us.
Electrostatic Warning

The processor board in the kit is sensitive to static electricity. Handle with care and be careful to discharge yourself before handling. Use the silver anti-static bag as an anti-static mat. Touch the bag before handling the processor to reduce the risk of damage.

Work on a flat surface, not the bed, a carpeted floor, other anywhere where static electricity can build up. The warranty does not cover damage due to electrostatic discharge. Fall and winter are times when electrostatic electricity is more prevalent.

Discharge yourself by touching a metal door frame, a metal lamp that is properly grounded, metal furniture, or some metal structure that is not connected to power.
Table of Contents

- Overview
- Kit Assembly
- Experiments
- Troubleshooting (last four pages)
Parts List

- Electronics plate
- Processor board
- Solderless breadboard
- 8 4-40 \( \frac{1}{2} \) inch screws
- 12 4-40 Kep nuts
- AA Battery holder
- 10 Kohm resistor
- 4.7 Kohm resistor
- 4 270 ohm resistor
- 10 jumpers
- USB cable

- Red LED
- Yellow LED
- Green LED
- Infrared LED
- IR receiver
- Speaker
- Ultrasonic range sensor
- Thermistor
- Light sensor
Overview
The developer kit contains a processor board that is Arduino compatible using an ATMEGA328, a solderless breadboard, an electronics plate and components and jumper wires.

The kit is designed to be modular so it can be more easily integrated.
The processor board is the brains of the kit. It has 32 Kbytes of program memory and 2 Kbyte of data memory. It operates at 16 MHz.

The processor board has multiple interfaces

- Digital signals
- Servo motor
- Communications
- Analog inputs
Electronics Plate

- This is a fiberglass board that is used to hold the processor board and solderless bread board.
Solderless Bread Board

- The solderless bread board allows you to wire up circuits. It will be used to connect sensor and components together and to the processor board.
How the Solderless BreadBoard Works

- The solderless bread board allows circuits to be quickly connected.
- Each row of holes that go left to right on the top and bottom are all connected together.
- The columns of 5 holes are all connected together.
- The lines in the picture show the connections.
- Components and wires are inserted in the holes to make connections.
Electronics Base Plate

• Base plate is for mounting solderless bread board and processor board
• Solderless bread board is to be mounted in the marked rectangular area.
  • Use the 1/2 inch screws and nuts to secure as shown in the next page.
  • Insert the screws from the top through the solderless bread board and secure with 4-40 nuts on the back side.
  • Make sure the solderless bread board is oriented as shown in the picture on the next page.
Mount the Solderless Breadboard

- The solderless breadboard is mounted as shown.

½ inch 4-40 Screws
The process board is mounted differently.
- Insert screws from the back side and install a nut on each screw. The nuts will serve as standoffs for the processor board. Look at the picture what holes are used.

½ inch Screws with Nuts
Electronics Base Plate Assembly

- Place the processor board on top of the nuts.
- Insert another set of four nuts to secure the processor board.
Processor Board and Arduino Software
Overview

- In this section, you will be introduced to the processor board electronics and the arduino software.
- At the end of this section, you will be able to write software, control things and sense the environment.
Processor Specifications

- The processor board is shown to the right. It is called an embedded computer because it is to be integrated or embedded in something.

- The processor board has connections that allow devices to be interfaced such as lights, motors and sensors.

- There are two types of interfaces that will be used, digital and analog.

- The digital interfaces can be configured as an input or output.
  - As an input, the digital interface can detect the state of switches or other signals as being on or off.
  - As an output, the digital interface can turn things on and off such as lights.

- The analog interface is an input only can can measure voltages.
The processor board uses an ATMEGA328. It has 32 Kbytes of program memory and 2 Kbyte of data memory. It operates at 16 MHz.

The processor board has multiple interfaces:

- Digital signals
- Servo motor
- Communications
- Analog inputs
The digital pins are highlighted in green.

There are six digital pins identified as D3, D5, D6, D9, D10, D11 which can also be used to control servo motors.

Adjacent to those pins are additional digital pins D2, D4, D7, and D8.

The digital pins can be used to control things and detect things.

When the digital pin is set low, the voltage is set to zero volts.

When the digital pin is set high, the voltage is set to 5 volts.
The digital pin configuration is shown to the right.

The digital pin is the most inward pin. This is the digital signal that can be set to high or low, logic level 1 or 0, 5 volts or 0 volts.

The next pin is 5 volts. This is power that can be used. It is available when the power switch is set to on and an external power source is connected. The USB port does not power this 5 volts for the protection of the host computer.

The last pin closest to the edge of the board is the ground. This is the reference voltage of zero volts.

The pins are positioned so a servo motor can be directly plugged in.

Each pin can be configured as an input or output.
Power Selection

- There is a three pin jumper that lets you select how the processor board is powered.
- A shorting jumper is required to make the selection.
- The picture below shows the shorting jumper connecting USB which connects the USB port to power.
- With the shorting block in the USB position, the USB port would power the processor eliminating the need for batteries.
Other Features

- With the power selection set to EXT, the terminal block is connected to power. The power switch only works with external power.
- With the jumper set to EXT, the USB port cannot power the processor board.
- The 5V power on the digital pins are only available from external power. This is done to protect the laptop from overloads and noisy servos if servos are used.
Other Features

- Between the terminal block and the power selection are two pins allowing access directly to battery power and ground.

- The BAT pin is connected to the Terminal block through the power switch. It does not give access to USB power.
Other Features

- The terminal block allows external power to be supplied to the processor board. To use external power, switch the shorting block from USB to EXT.

- Use no more than 12 volts.
Now that the processor features have been covered, it is time to learn about programming it.

The processor uses the arduino software. This software allows you to write programs, compile them and upload them to the processor. It also allows you to interact with the software running on the processor.

Only one program can be installed and run at a time. The processor is small and does not have an operating system.

Embedded computers are designed to perform a specific task and not operate like a desktop computer or laptop.

More information about the arduino software can be found at

- www.arduino.cc
Loading and Configuring Arduino Software

- Download the latest version of Arduino software from www.arduino.cc
  - It can be installed anywhere on the computer.
  - A version is available for Windows, MAC and Linux.
  - Open the folder and double click Arduino.
- The first step is to select the correct processor. Arduino software supports many different variations.
- For Arduino software versions earlier than 1.6
  - In the arduino program select menu “Tools”
  - Select “Board”
  - Select “Arduino Pro or Pro Mini (5V 16MHz) with ATMega328”
Loading and Configuring Arduino Software

- For Arduino version 1.6 and later
  - Go to the Tools menu and select Board.
  - Under the Board menu, select Arduino Pro or Pro Mini
  - Next under the Tools menu, select Processor
  - Under the Processor menu, select Atmega328 (5V, 16 MHz)
Selecting COM Port

- The Arduino software communicates with the processor board for uploading code and interaction via a COM port with Windows and a /dev/tty.usbserial-Daxxxxxx with MAC OS X. Under Linux, the COM port is /dev/ttyUSBx where x is a number.

- For Linux, the device driver is already part of the operating system.

- The next two pages explain how to load the device driver for Windows and MAC OS X. Administrative privileges are required for installation. Under Windows, you must run the installation software as an administrator. Under MAC OS X, you will be asked for your login password.
If the Serial Port menu does not show any COM ports try the following:

- Go to [http://www.ftdichip.com/Drivers/VCP.htm](http://www.ftdichip.com/Drivers/VCP.htm)
- In the table on the website, locate the row specifying Windows.
- Click on the link `setup executable` and download the software.
- Right click on the icon labeled `CDM v2.12.00.....` and select Run as Administrator in the menu that pops up.
- Follow the instructions to install. You have to run the installation program as an administrator or the driver will not install.
- Go back to Arduino and select the COM Port it gives you.
  - Most likely, the COM port will be COM3. If there is more than one COM port, choose the higher number.
Macintosh OS X USB Driver Installation

- Go to http://www.ftdichip.com/Drivers/VCP.htm
- Select VCP driver (Virtual COM Port) for Mac 64 bit 2.2.18
- Download and double click on the file
  - A window will open showing two packages
- Double click the package for the OS version you have
- Follow instructions for the installation
- Go back to Arduino and select the Serial Port: /dev/tty.usbserial-DAxxxxxx
  - The xxxxxx will be some combination of letters and numbers
Using Arduino

- This is the Arduino software.
- The software will let you enter programs and upload the code to the processor board.
- The large white area is where the code is entered.
- The black area below is where error messages will be displayed such as when there is an error in the code or the software cannot upload code for some reason.
Using Arduino

- The buttons below the menu have different functions.
- The first called Verify Code will compile the code and check for errors but not upload the code.
- The next button will do the same as the first but also upload the code.
- New Program button opens a new copy of the program allowing you to start writing another program.
- Open and Save are for opening and saving the code you have written.
Using Arduino

- Serial Monitor button opens a new window allowing you to interact with the processor.
- The Serial Monitor window allows the processor to display information and you to send information.
- This will be used quite a bit in this section.
Power Selection for Programming

- Before continuing, the power selection shorting jumper needs to be moved to the USB side.
- Pull the shorting jumper from the two pins and insert where it is marked USB.

Move jumper to left side.
First Program to Test

- Enter the program in the editor on the right. **Do not copy and paste from the pdf file.** It doesn't work. The compiler is case sensitive so pay attention to capitalized letters.

- Plug the processor board into the USB port.

- Click on the upload Code button to compile and upload the program.

- When the status message at the bottom of the window says done uploading, click on the serial monitor button.

- The Serial Monitor window pops up with the message being displayed.

- Experiment by changing the message.

- Save your program. Pick a file name.

```c
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
}
```
What are Functions

- A function is basically a set of instructions grouped together. A function is created to perform a specific task.

- The set of instructions for a function are bounded by the curly brackets as seen to the right.

- The `setup()` function is used to initialize the processor board, variables, and devices.

- Inside functions, you can call other functions. `Serial.begin()` is a function. It is located somewhere else in the arduino software.

```cpp
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
}
```
Other Syntax Requirements

- You will notice that some lines end with a semi-colon. This is used to identify the end of an instruction. An instruction can be an equation or function call.

- When you create a function such as `setup()`, you do not need a semi-colon.

```cpp
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
}
```
Arduino Programming Basics

- The program is made up of two functions.

- **setup()** function is run at reset, power up or after code upload only once.
  - It is used to initialize all the needed interfaces and any parameters.

- **loop()** function is run after the **setup()** function and is repeatedly run hence the name loop.

- This program configures the serial interface to send messages at 9600 bits per second.

- The message is “Hello World” and is repeatedly displayed.

```plaintext
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
    delay(500);
}
```

- **Serial.begin()** is a function that initializes the serial interface and sets the bit rate.
- **Serial.println()** sends the specified message over the serial interface and move the cursor to down one line.
- **delay(500)** is a command to stop the program for 500 milliseconds.
What is in the Software

- In the `setup()` function, it executes the function `Serial.begin(9600);`
  - This function initializes the UART which is connected to the USB port to allow for communications.
- In the `loop()` function, it executes the function `Serial.print(“Hello world”);`
  - This function send the text in quotes to the UART. This is displayed in the Serial Monitor window.
- The other function is called `delay()`.
  - This function stops the program for a specified period of time. The unit is in milliseconds. The code to the write displays the text every half second.
What is in the Software

- In the Serial Monitor window, you may have noticed that the text displayed scrolls to the right. That is just how `Serial.print()` works.

- To have the text displayed on its own line, change the `Serial.print()` to `Serial.println()`.

- `Serial.println()` adds a line feed which forces the text in the Serial Monitor to move down one line.

- Make the change, upload the code and open the Serial Monitor window.

```cpp
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.println("Hello World");
    delay(500);
}
```
The processor board has an LED, or light emitting diode. It is connected to digital pin 8.

The code to the right makes the LED blink. Try it out.

Once it works, adjust the delays to change how it blinks.

The next section will describe the `digitalWrite()` function.

```cpp
void setup()
{
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(8, HIGH);
  delay(500);
  digitalWrite(8, LOW);
  delay(500);
}
```
Electronics

- At this point, you should be able to run the Arduino software.
- You should know that the software consists of two functions
  - `setup()`
  - `loop()`
- You should know how to initialize the UART and write a program to display text in the Serial Monitor window.
- You should know how to open the Serial Monitor window.
- Next is learning a little about electronics and how to control things.
Electrical Circuits

- In this section, you will learn how to connect electrical circuits.
- Electrical circuits is nothing more than connecting wires between devices to allow the flow of electrons. A lamp plugged into an outlet has two wires to make an electrical circuit.
How the Solderless BreadBoard Works

- The solderless bread board allows circuits to be quickly connected.
- Each row of holes that go left to right on the top and bottom are all connected together.
- The columns of 5 holes are all connected together.
- The lines in the picture show the connections.
- Components and wires are inserted in the holes to make connections.

Back side showing how holes are connected
First Circuit

- The first circuit will use a Light Emitting Diode or LED.
- The LED is a polarized device and only works in one direction and gives off light when current flows through it.
- The positive pin on the LED is the longer pin. It is called the anode. The other end is the cathode.
- LEDs need the current to be limited otherwise it will take too much and burn out. A resistor will be used to limit the current flow through the LED.
How Resistors Work

- The popular analogy to describe a resistor is to compare it to plumbing.
- The water pipe is the wire and a resistor is a section of pipe that is narrower than the other pipes. This restricts how fast water can flow. The smaller the section of pipe the more resistance.
The first circuit will connect the LED straight to 5 volts so the LED will always be lit when there is power.

The schematic for the circuit is shown to the right.

- The symbol at label R1 is for the resistor.
- LED1 is next to the symbol for the LED.
- The symbol at the top is the +5V connection. It is called VCC.
- The GND symbol is for ground. This is the zero volt reference.
- The LED has an anode and a cathode. The anode is usually the long pin.
  - Not all manufacturers follow this.
- When the anode is at a higher voltage than the cathode, the LED will light.
A resistor is a device that is used to limit the flow of current or reduce the voltage depending on how it is used.

Resistor values are determined by the color bands. The picture to the left shows how to decipher the color bands. The first two bands determine numerical value and the third band is the multiplier. A 270 ohm resistor has a red, violet, and brown band. The last band indicates the how far off the value can be.

The bands start at one end of the resistor.
Insert the LED into the bread board as shown. The short lead on the LED should be on the left side.

Insert the 270 ohm resistor as shown. One lead should be installed in the same column as the long lead of the LED. The other resistor lead is inserted in any other column.

Take the black jumper wire and insert the pin into the column of the short LED lead. Plug the other end into the ground pin as shown.

Take a red jumper and connect the pin into the same column as the resistor lead and the other end into 5V pin as shown.

Plug the processor board into the computer USB port. The LED should light up. If not, reinsert the LED in the opposite orientation.
LED Connected to Digital Pin 3

Move the red jumper from the 5V pin on the processor board to the pin marked D3.

Leave everything else as is.
Connecting the LED to a Digital Pin

- The LED is not lit at this time because the digital pin 3 needs to be programmed to generate a voltage.
- The program to the right will cause the LED to blink.
- Create a new program and enter the code.
- Upload the code to the processor board. Make sure the processor board is plugged into the USB port.
- Save the program and use a new file name such as “blinky”.

```cpp
void setup()
{
    pinMode(3, OUTPUT);
}

void loop()
{
    digitalWrite(3, HIGH);
    delay(500);
    digitalWrite(3, LOW);
    delay(500);
}
```
Connecting the LED to a Digital Pin

- In the `setup()` function, digital pin 3 is configured as an output.
  - The function `pinMode()` configures digital pin 3 to be an output.
  - `pinMode()` takes two arguments separated by a comma.
    - The first argument selects the digital pin.
    - The second argument configures the digital pin as an output.
- In the `loop()` function, digital pin 3 is set high which causes the pin to generate 5 volts. The LED turns on.
- The `delay()` function halts the program for 500 milliseconds.
- The next `digitalWrite()` command sets digital pin 3 to 0 volts turning off the LED.
- Save the program and use a new file name such as “blinky”.

```cpp
void setup()
{
  pinMode(3, OUTPUT);
}

void loop()
{
  digitalWrite(3, HIGH);
  delay(500);
  digitalWrite(3, LOW);
  delay(500);
}
```
The `digitalWrite()` function controls a pin and can set it high or low.

- The function has two arguments separated by a comma.
  - The first argument selects the digital pin.
  - The second argument sets the digital pin.
  - When set high, the pin is set to 5 volts.
    - In logic terms, a high signal is logic level 1.
  - When set low, the pin is set to 0 volts.
    - In logic terms, a low signal is logic level 0.
- The function is written as
  - `digitalWrite(pin,setting)`
    - setting is HIGH or LOW
      - The letters need to be capital.

```
HIGH   = 5 volts = Logic 1
LOW    = 0 volts = Logic 0
```
Using the Speaker

- The speaker is polarized.
- Look at the bottom for the “+” next to the pin.
- The “+” pin is to be connected to the digital pin. Try digital pin 3.
- The other pin connects to ground.
- The tone command generates a square wave at the specified frequency on the specified pin.
- The tone commands uses two parameters.
  - The first parameter is the pin number.
  - The second parameter is the frequency in hertz.

```cpp
void setup()
{
    pinMode(3, OUTPUT);
}

void loop()
{
    tone(3, 2000);
    delay(500);
    tone(3, 3000);
    delay(500);
}
```
There is a simple circuit that uses two resistors to divide a voltage from a higher level to a lower level.

Voltage dividers can be used to reduce a voltage that is too high to a lower voltage that can be handled.

Another use for this type of circuit is if one of the resistors is a sensor like temperature or light sensor where its resistance changes with what it measures. This circuit makes it simple to connect to an analog-to-digital converter.

The voltage output, $V_{out}$, is given by the formula:

$$V_{out} = +V \times \frac{R2}{R1+R2}$$
Resistor Voltage Divider

- The resistors are connected together in series.
- The voltage source which is the battery in the picture to the left is connected across both resistors.
- The divided voltage is located at the connection between the two resistors.
- The voltage can be calculated by the equation to the left.
  - $V$ is the voltage across both resistors.
  - $R_1$ and $R_2$ are the resistors of some value. They can be any value to get a variety of different voltages.

\[ V_{\text{out}} = V \times \frac{R_2}{R_1 + R_2} \]
Analog-to-Digital Converter

- An analog-to-digital converter or ADC is a device that generates a number based on the voltage level it measures.
- The ADC on the processor board can measure a voltage range from zero to 5 volts. If a higher voltage needs to be measured, the voltage divider circuit could be used to reduce the voltage to 5 volts or less.
- The ADC is 10 bits. This gives a numerical range of zero to 1023. It is a linear relationship to the voltage range of 0 to 5 volts.
  - The ADC will generate a value of 0 for 0 volts, 1023 for 5 volts, and 511 for 2.5 volts. The resolution of the ADC is 5/1023 or 0.00489 volts per bit.
  - Voltage the ADC measures can be calculated by the equation
    \[ V = \frac{\text{ADC value}}{1023} \times 5.0 \]
The analog ports are highlighted in yellow.

Six analog inputs are available: 0,1,2,3,6,7.

The analog ports allow the measurement of voltages from sensors that generate a voltage based on what is being measured.

Each analog port has 10 bit resolution and an input range of 0 to 5 volts. Signals beyond this range need to be converted to operate within the 0 to 5 volts or damage may occur.

The processor has an analog to digital converter or ADC to convert the analog voltage to a digital value. The ADC has a 10 bit resolution which converts 0 to 5 volts to a number with a range of 0 to 1023. It is a linear relationship.

To calculate the voltage

\[ \text{voltage} = \frac{\text{ADC}}{1023.0} \times 5.0 \]
The photo cell is a light sensitive device that changes its resistance based on light intensity.

The photocell can be used in a simple voltage divider circuit with a 4.7Kohm resistor. The color code is yellow, violet and red.

The photo resistor will have a resistance ranging from 20 Mohm in darkness to 5K ohms in bright light.

Install the photo cell and 4.7 K resistor on the solderless bread board as shown to the right.

Connect the free end of the resistor to GND at the analog connector.

Connect the free end of the photo cell to 5 volts.

Connect the resistor and photo cell connection to pin 0 of the analog connector.

To the right is the schematic.
The program to the right will get an ADC value from analog port 0.

Create a new program and enter the program to the right.

To measure the voltage, the function `analogRead(port)` is used.

Six ports are available on the processor board.
- 0, 1, 2, 3, 6, 7
- Refer to page 5 for the location.

Once the ADC value is read, it can be converted to a voltage value. The code to the right shows the equation which can be used for all the analog ports.

The `Serial.println` function that displays the volts, includes a numeric argument which specifies the number of decimal places.

Save the program. Use a new name like `photocell`.

```cpp
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0 * 5.0;
  Serial.println(volts,2);
  delay(200);
}
```
Temperature Sensor

- The temperature sensor shown here is called a thermistor. It is a device that changes its resistance based on temperature. The thermistor will use the same type of voltage divider circuit as the light sensor. The resistance is not quite linear with temperature but can be approximately linear in a small temperature range.

- The table to the right shows the resistance at specific temperatures.

- This thermistor has a 10 Kohm resistance at 25 C.

<table>
<thead>
<tr>
<th>Temperature C</th>
<th>Resistance Ohms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32544</td>
</tr>
<tr>
<td>5</td>
<td>25339</td>
</tr>
<tr>
<td>10</td>
<td>19872</td>
</tr>
<tr>
<td>15</td>
<td>15698</td>
</tr>
<tr>
<td>20</td>
<td>12448</td>
</tr>
<tr>
<td>25</td>
<td>10000</td>
</tr>
<tr>
<td>30</td>
<td>8059</td>
</tr>
<tr>
<td>35</td>
<td>6535</td>
</tr>
<tr>
<td>40</td>
<td>5330</td>
</tr>
<tr>
<td>45</td>
<td>4372</td>
</tr>
<tr>
<td>50</td>
<td>3605</td>
</tr>
</tbody>
</table>
Thermistor

- Assemble the circuit as shown. Use Analog port 0 just like the photocell.
- Instead of using a 4.7 Kohm resistor, a 10 kohm resistor will be used.
- At 25°C, the voltage at analog port 0 will be 2.5 volts.

```c
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int a;
    float volts;
    a = analogRead(0);
    Serial.println(a);
    volts = (float)a/1023.0 * 5.0;
    Serial.println(volts,2);
    delay(200);
}
```
• Knowing the resistance of the thermistor at specific temperatures, calculate the voltage at the temperatures and modify the program to the right to display the temperature.

• The tolerance of the thermistor is 3%.

```cpp
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0 * 5.0;
  Serial.println(volts,2);
  delay(200);
}
```
Ultrasonic Range Sensor

- The ultrasonic range sensor is a device that sends a short burst of sound and listens for the echo.
- The processor board starts the measurement by generating a pulse on the Trig pin.
- It then measures the size of the pulse on the Echo pin.
- The processor calculates the distance based on the size of the pulse width.
- Distance is calculated by dividing the pulse width measured in microseconds by 58. Answer is in centimeters.
Ultrasonic Range Sensor Operation

- The ultrasonic range sensor operates in a specific sequence.
- It waits for a trigger signal. The trigger is a 10us pulse. Once the trigger is detected, the sensor generates a short signal at 40 KHz.
- It then waits for an echo and measures the time from sending the short burst to receiving the echo.
- The sensor then generates a pulse on the echo with a length proportional to the delay measured.

\[
\text{distance} = \frac{\text{pulse width (us)}}{58}
\]
Ultrasonic Range Sensor

- Insert the ultrasonic ranger as shown. It should be mounted close to the center of the robot. The pins are inserted at the end of the rows.

- Connect jumpers from the sensor to the processor
  - GND to Analog GND
  - ECHO to pin D3
  - TRIG to pin D5
  - VCC to Analog 5V

- Look on the processor board for the word ANALOG. The power connections are done there to isolate the sensor from the motor power to reduce electrical noise.
The ultrasonic sensor has two signals, trigger and echo.

A pulse is sent to the trigger and then the processor is to time when the echo returns.

This requires two digital pins, one configured as an output and the other as an input. A new command that will be used is called `pulseIn()`. This measures the time it takes a pulse to occur in microseconds. Try the program to the right.

The results are in centimeters.

Create a new program and enter the code to the right. Save the program and upload it.

```cpp
void setup()
{
    Serial.begin(9600);
    pinMode(3, INPUT);
    pinMode(5, OUTPUT);
}

void loop()
{
    unsigned long distance;
    digitalWrite(5, LOW);
    delayMicroseconds(2);
    digitalWrite(5, HIGH);
    delayMicroseconds(10);
    digitalWrite(5, LOW);
    distance = pulseIn(3, HIGH);
    distance = distance / 58;
    Serial.println(distance);
    delay(500);
}
```
Creating a Separate Function File

1. Start a new program with the Arduino program.
2. Click on the down arrow to the right where circled in red.
   - A menu will open. Select “New Tab”
   - Below, it will ask for a name. Enter 'ultrasound'
     - Do not put .c on the end of the name or the program will not compile properly.
3. Click 'OK'
4. A new tab is created called 'ultrasound'
5. Type in the ultrasonic function.
Making a Function

- To make this useful for other programs, this program needs to be turned into a function.

- A function is a subroutine or chunk of code that can be called by a name instead of the code being inserted where ever it is needed. This function will return a result.

- The return command specifies which variable is sent back to the calling code.

```c
unsigned long ultrasonic()
{
    unsigned long distance;
    digitalWrite(5, LOW);
    delayMicroseconds(2);
    digitalWrite(5, HIGH);
    delayMicroseconds(10);
    digitalWrite(5, LOW);
    distance = pulseIn(3, HIGH);
    distance = distance/58;
    return(distance);
}
```

The function `pulseIn()` returns the number of microseconds. The result is then divided by 58 to calculate the distance in centimeters.
Creating a Separate Function File

- Click on the first tab.
- Enter the program to the right.
- Upload it and run it.
- The tabs allow you to organize your code better by separating functions from the main part of the code.

```cpp
void setup()
{
  Serial.begin(9600);
  pinMode(3, INPUT);
  pinMode(5, OUTPUT);
}

void loop()
{
  unsigned long distance;
  distance = ultrasonic();
  Serial.println(distance);
  delay(500);
}
```
Infrared Remote Control

- Find a TV remote. Chances are very good that it uses an infrared LED to send signals to control the TV. The remote will be used to control the robot.
- First, the control codes need to be captured. The first program will do that.
- There are two ways remotes work. One way is for the remote to resend the code for the key pressed repeatedly until the key is released. The second is to send the code for the key pressed then send a code that equals all '1's.
- The next program will be used to capture the codes to be inserted into the robot program. The program decodes the key code and displays it in hexadecimal.
- First, remove the Ultrasonic sensor from the solderless bread board.

- NOTE: If you are using Arduino software version 1.0.6 or greater, you need to delete a library from the software. Locate the libraries folder in the Arduino software and delete RobotIRremote library. It interferes with the IR library used with this material.
How IR Remotes Work

- The IR remote uses an LED that operates in the infrared range, specifically 990 nanometers. The emitter pulses the infrared light at 38 KHz, 38,000 times per second. The pulsed signal is then turned on and off at a lower rate so that bursts of 38 KHz light is transmitted. The LED is modulated to help the receiver detect the signal from other light in the room including sun light.

- The turning on and off of the modulated light is done in different sequences to generate different codes.
The infrared Receiver is a device that includes an IR detector and a circuit to detect IR signals modulated at 38 Khz.

A photo diode is the sensor and connects to an input circuit that converts the current to a voltage.

The signal goes through a bandpass filter. This is a filter that only lets a signal of a specific frequency to pass. All other signals cannot pass.

The filtered signal is then sent to a demodulator that converts the modulated signal to the codes being sent.
Add the IR receiver to the solderless breadboard. Note the orientation as the connections need to be made in a specific order or the receiver may be damaged.

Look at the picture to the right of the IR receiver. The leads on the IR receiver are numbered in a specific orientation.

When mounting on the solderless bread board, have the rounded side face forward.

Pin 1 – OUT
Pin 2 – 5V
Pin 3 - GND
Wiring the IR Sensor

- Connect pin 3 of the IR sensor to GND under Digital pin D3.
- Connect pin 2 of the IR sensor to 5V Analog.
- Connect pin 1 of the IR sensor to Digital pin D3.
- Be careful to not connect power backwards or the IR sensor will be damaged.
Getting the IR Library

- A library that allows the decoding of infrared remotes such as TV remotes is available for the Arduino software.
- Go to the website shown below. Along the right side of the website is a button labeled “Download ZIP”.
- Follow the instructions in the readme on the website.

https://github.com/shirriff/Arduino-IRremote
Remote Code Capture Program

- In the Arduino software, select File then Examples. Look for Irremote and select IRrecvDemo.
- When the program is loaded, look for the line 'int RECV_PIN = 11;' and change the pin number to 3.
- Compile and upload to the processor board.
- Open the Serial Monitor.
- Now point a remote control at the IR receiver and press a button.
- Numbers should be displayed in the serial monitor. Values will be in hexadecimal.
- Remotes operate in one of two ways. One type of remote keeps repeating the code for the button pressed as long as the button is pressed. The other type will send the code once and then all logic level 1 after that.
- The codes are generally 32-bit numbers. The program displays the numbers in hexadecimal. There should be 8 digits. There are some remotes that will send smaller codes.
Capturing the Remote Codes

• With the program running, press three buttons and record the codes generated. These codes will be used in the next program.

  Button 1 ________________
  Button 2 ________________
  Button 3 ________________
Robot Control Program

- The next program will receive the IR commands and activate an LED for the selected button.
- The sequence is to receive a key code, compare it, then execute the proper code then go back and wait for the key code to be sent again.
- Connect the three LEDs to Digital pins 6, 9, 10. Don't forget to include the 270 ohm resistor.
IR Remote Code

- Start a new program.
- Under the **Sketch** menu, select the **Include Library** and then select **Arduino-IRremote-master** library.
- This will insert an include file statement at the top of the program. Two are inserted. Delete the one not shown to the right.
- Enter the program to the right and next page.
- Replace the underlined words with the appropriate code. Since the numbers are in hexadecimal, insert **0x** in front of the number.

```cpp
#include <IRremote.h>
unsigned long tt;
IRrecv irrecv(3);
decode_results results;

void setup()
{
  irrecv.enableIRIn();
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
}
```
void loop()
{
  int tt = 0;
  if(irrecv.decode(&results)) {
    if(results.value == **Button 1**) {
      digitalWrite(6,HIGH);
    } else if(results.value == **Button 2**) {
      digitalWrite(9,HIGH);
    } else if(results.value == **Button 3**) {
      digitalWrite(10,HIGH);
    }
    tt = 0;
    irrecv.resume();
  }
  tt++;
  if(tt > 3400) { // increase value if LEDs flicker
    digitalWrite(6,LOW);
    digitalWrite(9,LOW);
    digitalWrite(10,LOW);
    tt = 0;
  }
}
IR Remote Code (continued)

- Remote controls repeat the associated code as long as the button is pressed. The rate varies among remotes.

- The `tt` variable increments between when the code is transmitted. If it stops, `tt` exceeds a limit and that is a condition to turn off the LEDs.

- Change the value 3400 to a different number to see what happens.

```c
void loop()
{
    int tt = 0;
    if(irrecv.decode(&results)) {
        if(results.value == Button 1) {
            digitalWrite(6,HIGH);
        } else if(results.value == Button 2) {
            digitalWrite(9,HIGH);
        } else if(results.value == Button 3) {
            digitalWrite(10,HIGH);
        }
        tt = 0;
        irrecv.resume();
    }
    tt++;
    if(tt > 3400) { // increase value if LEDs flicker
        digitalWrite(6,LOW);
        digitalWrite(9,LOW);
        digitalWrite(10,LOW);
        tt = 0;
    }
}
```
• The IR LED connects and operates like a regular visible light LED.

• The IRRemote library requires the LED be connected to digital pin 3. It cannot be reassigned.

• This experiment will require one Arduino board set up to receive the IR signal and running the IrrecvDemo program and another Arduino set up to send.
Once the object IRsend is declared, it is automatically created. IRsend uses digital pin 3 only.

Even though the function setup() is empty, it still needs to be created in order for the software to work.

The loop includes a 40 ms delay between transmissions. This is required so the receiving end can recognize the end and beginning of a transmission.

The Arduino running IRrecvDemo should display the value in hexadecimal, 12345678.

```cpp
#include <IRremote.h>

IRsend irsend;

void setup()
{
}

void loop()
{
    irsend.sendSony(0x12345678,32);
    delay(40);
}
```
The function `irsend.sendSony()` takes two arguments. The first is the bits represented in hexadecimal and the second is the number of bits to send.

A maximum of 32 bits can be transmitted at a time.

```c
#include <IRremote.h>

IRsend irsend;

void setup()
{
}

void loop()
{
    irsend.sendSony(0x12345678,32);
    delay(40);
}
```
• Variables can be sent instead of fixed numbers. The code to the right shows how it is done. The code increments a variable from 0 to 99,999 and sends the value.

• The receiver should receive the signal and display an incrementing count.

```cpp
#include <IRremote.h>

IRsend irsend;

void setup()
{
}

void loop()
{
  long a;
  for(a=0;a<1000000;a++) {
    irsend.sendSony(a,32);
    delay(40);
  }
}
```
Other Useful Things
User Input

- To get an input from the user, the program to the right can be used.

- `Serial.available()` returns the number of characters received. If there are no characters received, the result is zero.

- `Serial.read()` reads a character sent to the processor board. It is assigned to an integer variable.

- The `Serial.print` statements print out what was received.

- You will notice that the variable `a` in the `Serial.println()` statement has `(char)` in front. This is type casting converting the integer into a character type variable. This makes sure the ASCII character is printed. Try without the `(char)` type cast and see what is the result.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Press a key");
  while(Serial.available() == 0);
  int a = Serial.read();
  Serial.print("The key pressed is: ");
  Serial.println((char)a);
}
```
Arduino provides a function to enter more than one character at a time.

Try the program to the right and see how it behaves.

Serial.readBytesUntil() is a bit primitive. You will notice old parts of what was entered is still being shown with parts overwritten. Next page shows how to properly read in a string of characters.

```c
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    char buf[64];
    Serial.print("Enter your name: ");
    while(Serial.readBytesUntil(\r, buf, 64)==0);
    Serial.print("You typed : ");
    Serial.println(buf);
}
```
A Better Way to Receive Strings

- A function is created to read in a string of characters and stopping when the enter key is pressed which is the carriage return character. In C, the character is represented as '\r'.

- The function requires a parameter which is the character array name.

- In C, when passing an array to a function, the function receives a pointer to the array. The pointer is identified with '*'.

```c
int get_string(char *a)
{
    int i = 0;
    int c = 0;
    while(c != '\r') {
        while(Serial.available() == 0);
        c = Serial.read();
        i++;
        *a++ = (char)c;
    }
    *a = 0;
    return(i);
}

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    char buf[32];
    int count;
    Serial.print("Enter something:");
    count = get_string(buf);
    Serial.print("This is what was entered:");
    Serial.println(buf);
    Serial.print("The number of characters: ");
    Serial.println(count,DEC);
}
Conclusion

- You now have the basics for developing new things with the kit.
- For more tutorials and reference materials, go to www.arduino.cc
Processor Board Schematic
Troubleshooting

- This section covers common issues
  - No COM port found in Windows
  - USB Serial port for Macintosh
  - Uploading code not working
• If the Serial Port menu does not show any COM ports try the following:
  • Go to http://www.ftdichip.com/Drivers/VCP.htm
  • In the table on the website, locate the row specifying Windows.
  • Click on the link setup executable and download the software.
  • Right click on the icon labeled CDM v2.12.00..... and select Run as Administrator in the menu that pops up.
  • Follow the instructions to install. You have to run the installation program as an administrator or the driver will not install.
  • Go back to Arduino and select the COM Port it gives you.
    • Most likely, the COM port will be COM3. If there is more than one COM port, choose the higher number.
Macintosh OS X USB Driver Installation

- Go to [http://www.ftdichip.com/Drivers/VCP.htm](http://www.ftdichip.com/Drivers/VCP.htm)
- Select VCP driver (Virtual COM Port) for Mac 64 bit 2.2.18
- Download and double click on the file
  - A window will open showing two packages
- Double click the package for the OS version you have
- Follow instructions for the installation
- Go back to Arduino and select the Serial Port: /dev/tty.usbserial-DAxxxxxx
  - The xxxxxx will be some combination of letters and numbers
Problems Uploading Code

• Here are common problems uploading code
  • Make sure the processor board is powered. If powered from USB, make sure the power selection jumper is in the USB position. See page 22.
  • Make sure the proper processor is selected. It does make a difference.
    Arduino Pro or Pro mini 5V,16Mhz w/ATMega328
  • Make sure the proper COM or serial port is selected.
  • If there are other devices or components attached to the UART signals, disconnect them. They can interfere with the uploading.
  • It is also possible that the Arduino software is having a problem. It uses a program called avrdude to upload the program. Sometimes the program hangs and interferes with any attempts to upload. Either reboot the computer with the arduino software or if you know how, find the program avrdude and stop it using task manager under windows or the kill command under Mac OSX or Linux.
  • Last resort, reboot the computer.