Demonstration of one way to create a clock in Arduino

```
#include "ClockBasics.h" // Header that sets up the Clock

// ClockTimer
#define CLOCK_INTERVAL 1000
unsigned long ClockTimer;

// Simple timer to flash LED.
#define LED_INTERVAL 500
unsigned long LedTimer;

// Run once, To set up system
void setup()
{
	// Initialize LED timer and pin
	LedTimer = millis();
	pinMode(13, OUTPUT);

	// Serial Com.
	Serial.begin(9600);

	// Set up clock
	ClockTimer = millis();
	Hours = 23;
	Minutes = 59;
	Seconds = 55;

} // End of setup

// Code that is run continuously.
void loop()
{
	// LED Flashing Timer.
	if (millis() - LedTimer >= LED_INTERVAL)
	{
		// Toggle LED.
		if (digitalRead(13))
		{
			digitalWrite(13, LOW);
		}
		else
		{
			digitalWrite(13, HIGH);
		} // End of toggle

		LedTimer += LED_INTERVAL; // Update Timer.
	} // End of Led Timer.

	// Check for one second to update clock
	if (millis() - ClockTimer >= CLOCK_INTERVAL)
	{
		UpdateClock(); // move clock ahead one second.
		SendClock();   // send to serial monitor.
		ClockTimer += CLOCK_INTERVAL; //UPDATE TIMER
	}

	// Check for incoming serial data.
	if (Serial.available())
	{
		// Use incoming character to set clock.
		SettingClock(Serial.read());

	} // End of Serial available if

} // End of loop
```

Header file ClockBasics.h - Posted to "Files->CourseNotes->B_Programming->Support Code"

```
#ifndef ClockBasics_H
#define ClockBasics_H

// States for setting clock.
enum ClockStates { CLOCK_RUNNING, CLOCK_SET_HOURS, CLOCK_SET_MINUTES, CLOCK_SET_SECONDS };
ClockStates clockState = CLOCK_RUNNING;

// Variable used as clock value.
int Hours, Minutes, Seconds;
// This function is to be called every second to update the clock represented by the
// global variables Hours, Minutes, Seconds
void UpdateClock()
{
	// exit if clock being set.
	if (clockState != CLOCK_RUNNING)
		return;
	// Check if Seconds not at wrap point.
	if (Seconds < 59)
	{   Seconds++; // Move seconds ahead.
	}
	else
	{	Seconds = 0; // Reset Seconds
		// and check Minutes for wrap.
		if (Minutes < 59)
		{   Minutes++; // Move seconds ahead.
		}
		else
		{	Minutes = 0; // Reset Minutes
			// check Hours for wrap
			if (Hours < 23)
			{
				Hours++;// Move Hours ahead.
			}
			else
			{
				Hours = 0;// Reset Hours
			}// End of Hours test.
		} // End of Minutes test
	} // End of Seconds test
} // end of UpdateClock()
void SendClock()
{
	// Check if leading zero needs to be sent
	if (Hours < 10)
	{
		Serial.print("0");
	}
	Serial.print(Hours); // Then send hours
	Serial.print(":"); // And separator
	// Check for leading zero on Minutes.
	if (Minutes < 10)
	{
		Serial.print("0");
	}
	Serial.print(Minutes); // Then send Minutes
	Serial.print(":"); // And separator
	// Check for leading zero needed for Seconds.
	if (Seconds < 10)
	{
		Serial.print("0");
	}
	Serial.println(Seconds); // Then send Seconds
	// with new line
} // End of SendClock()
```

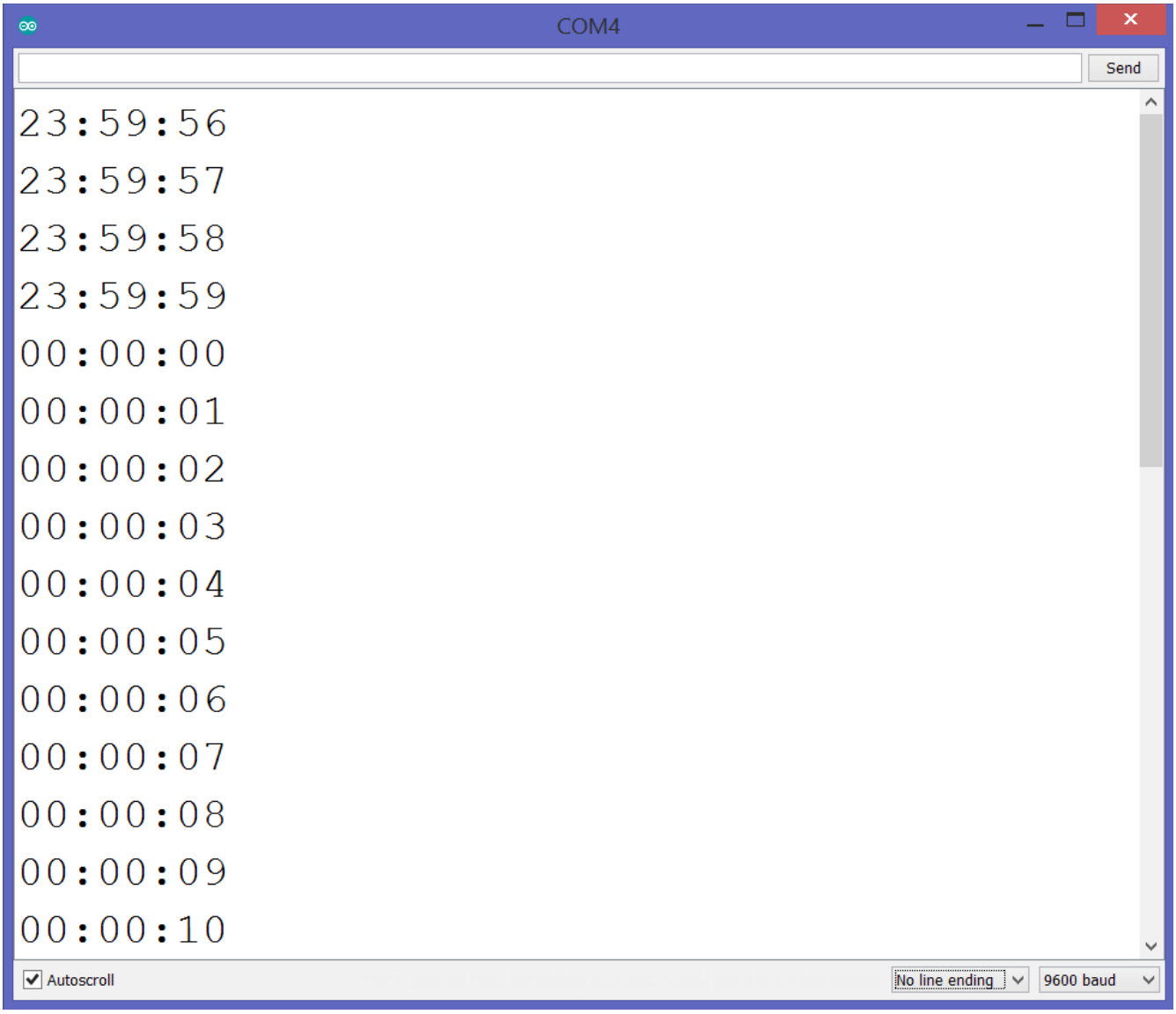
```
// Function that processes incoming characters to set the clock.
void SettingClock(char Input)
{
	// interpret input based on state
	switch (clockState)
	{
	case CLOCK_RUNNING:
		if (Input == 'S')
		{
			clockState = CLOCK_SET_HOURS;
			Hours = 0;   // Reset clock values.
			Minutes = 0;
			Seconds = 0;
		}
		break;
	case CLOCK_SET_HOURS: // Setting Hours
		if (Input >= '0' && Input <= '9') // if input is digit
			Hours = 10 * (Hours % 10) + Input - '0';
		else if (Input == ':')  // Move to next state
			clockState = CLOCK_SET_MINUTES;
		else if (Input == 'R') // Start clock
			clockState = CLOCK_RUNNING;
		break;
	case CLOCK_SET_MINUTES: // Setting Minutes
		if (Input >= '0' && Input <= '9') // if input is digit
			Minutes = 10 * (Minutes % 10) + Input - '0';
		else if (Input == ':') // Move to next state
			clockState = CLOCK_SET_SECONDS;
		else if (Input == 'R') // Start clock
			clockState = CLOCK_RUNNING;
		break;
	case CLOCK_SET_SECONDS: // Setting Seconds
		if (Input >= '0' && Input <= '9') // if input is digit
			Seconds = 10 * (Seconds % 10) + Input - '0';
		else if (Input == 'R') // Start clock
			clockState = CLOCK_RUNNING;
		break;

	}// End of clock mode switch.

} // End of SettingClock

#endif
```

Results as shown on the Serial Monitor



The following is the State Transition Diagram for the program that sets the clock.