2. Herzlich Willkommen zu unserer Präsentation. Im Rahmen unserer GFS sollten wir ein Auto bauen, das automatisch Hindernissen ausweicht und dabei die Unterschiede zwischen Arduino und Basic Stamp herausarbeiten

3. Wir beginnen mit den Unterschieden von Basic Stamp und Arduino. Danach erklären wir die wichtigsten Bauteile des Autos, das wir mithilfe eines Arduinos selbst gebaut haben. Am Ende lassen wir noch das Auto fahren.

4. Hier sehen wir einmal den Basic Stamp, den wir aus dem Unterricht kennen.

5. und hier einmal den Arduino, den ihr wahrscheinlich noch nicht kennt.

6. Beginnen wir mit den Unterschieden von Basic Stamp und Arduino. Der Basic Stamp besitzt einen Mikrocontroller der PIC16-Baureihe mit 4.0 MHz der Arduino aber besitzt einen mit 16 MHz und zwar den ATmega328P, , das bedeutet er ist 4mal schneller als der Basic Stamp. Beide geben 5V aus. Der Basic Stamp benutzt die Programmiersprache Basic (PBasic) der Arduino C oder C++. Der Basic Stamp hat 8 Pins die als Ein- oder Ausgang geschaltet werden können. Der Arduino hingegen hat davon 14, das heißt er bietet mehr Möglichkeiten. Außerdem hat der Arduino noch jeweils 6 analoge Eingänge und PWM-Pins, die der Basic Stamp nicht hat, wozu diese genutzt werden können erklären wir euch später noch. Kommen wir nun zu der Hilfe aus dem Internet. Über den Arduino findet man ziemlich viel da er sehr viel benutzt wird und weit verbreitet ist. Der Basic Stamp ist nicht so weit verbreitet und wird weniger benutzt, daher ist es schwer an Informationen zu kommen. Der Basic Stamp benutzt die Software Basic Stamp Editor, der Arduino benutzt Arduino Studio.

7. Nun zu den wichtigen Bauteilen des Autos. Wie erwähnt haben wir für das Auto den Arduino verwendet. Um die automatische Hinderniserkennung und das Ausweichen zu ermöglichen haben wir einmal den HC–SR04 als Abstandssensor benutzt der mit Hilfe von Ultraschall misst und für die Motorsteuerung haben wir den L293D benutzt.

8. Der Abstandssensor sendet Ultraschallwellen aus und misst deren Echo, das von Hindernissen zurückgeworfen wird. Die Abstandsmessung startet man indem man den Trigger Pin kurz auf high schaltet. Dadurch schickt der Sensor Ultraschalwellen los und der Echo Pin wird so lange high geschalten wie die Schalwellen für hin und zurück gebraucht haben. Der Mikrocontroller misst also die Zeit, die der Echo Pin high geschaltet ist. Da man in der Software einen Abstand vorgibt, bei dem das Auto ausweichen soll benötigt man noch die Entfernung. Diese wird über die Formel S = v\*t berechnet indem man die Schallgeschwindigkeit mit der gemessenen Zeit mal nimmt. Da die Zeit für den Hin- und Rückweg gemessen wurde wir aber nur eine der beiden Richtungen brauchen muss das Ergebnis noch halbiert werden.

9. Da man die beiden Motoren, die wir zur Steuerung des Autos benötigen nicht direkt an den Arduino anschließen kann, da diese zu viel Strom benötigen und dadurch der Mikrocontroller kaputt gehen würde, brauchen wir noch den L293D. Durch diesen integrierten Schaltkreis kann man entweder 4 Motoren jeweils nur in eine Richtung ansteuern oder 2 Motoren vorwärts und rückwärts. Zur Regelung der Geschwindigkeit des Autos haben wir die PWM Pins des Arduinos benutzt, durch die man die Geschwindigkeit eines Motors mit Hilfe von PWM, also Pulsweiten Modulation einstellen kann.

10. Das Signal hat immer die gleiche Frequenz, also die Summe der Zeit von an und aus ist immer gleich. Die Information für den Motor steckt darin wie lange das Verhältnis von an zu aus ist. Stehen bleiben wäre immer aus, also 0%. Die maximale Geschwindigkeit würde es erreichen, wenn es immer an wäre, also 100%.

11. Zum Schluss nochmal die für die Programmierung des Autos wichtigsten Befehle

1. Für den digitalen Ausgang nutzt man den Befehl digitalWrite (Pin, Wert), hier gibt man die Nummer des Pins und den gewünschten Wert high oder low an. Das ist der Befehl der beim Basic Stamp einfach High und Low wäre.   
Damit startet man z.B den Motor.

2. Für den digitalen Eingang nutzt man den Befehlt digitalRead (Pin), hier gibt man ebenfalls die Nummer des Pins ein, der das High oder low Signal einliest

3. Für die PWM Pins nutzt man den Befehl analogWrite (Pin, Wert) damit kann man die Geschwindigkeit eines Motors einstellen, indem man Werte von 0 bis 255 eingibt wobei 255 100% entspricht.

4. Für die Zeitmessung mit Hilfe des Echo Pins wird der Befehl pulseIn (Pin, High) genutzt.

12. und 13. Auf den folgenden Folien haben wir das Programm, das wir in der Programmiersprache C++ zur Steuerung unseres Autos geschrieben haben.

14. Hier noch die Quellen

15. Das war unsere Präsentation Jetzt kommen wir noch zum praktischen Teil und zeigen Euch unsere Autos.

Vielen Dank für die Aufmerksamkeit.